Personal Computer

PC-8801
NEC
mkⅡFR|MR

データファイル入門[1]

ナツメ出版企画＝編

ナツメ社

Personal Computer
PC-8801mkⅡFR|MR
データファイル入門[1]
ナツメ出版企画＝編
ナツメ社

[本書を読まれる方へ]

⦿パソコンをビジネスその他、
実務面で役立てようと思うと、
どうしてもディスク操作のマスターを迫られます。
しかしディスクでのデータの扱いは、
ベーシックの場合と多少考え方を異にするため、
すぐになじめない一面があります。
⦿これが、ディスクはむずかしいと言われる
ゆえんだと思いますが、この理由のために、
途中で挫折してしまうか、
あるいはせっかくディスクの備えがありながら、
これをプログラムのセーブとロードだけに
使用するにとどまっている人が多いようです。
⦿要するに、多くのディスクが
ユーザーの手に渡っていながら、
その本来の使命が十分果たされていないのです。
⦿そこで、ベーシックとディスクベーシックとの
連絡をつけ、みなさんができるだけ抵抗なく
入っていけるようにと考え、発行されたのが本書です。
ディスクは、あなたの努力次第で
ずいぶん実務面で活用できます。
⦿せっかくのディスクです。
なんとか活かしてみませんか？

cover design——nobuo nakagaki＋yoshifumi hayase

Personal Computer

PC-8801

NEC

mkII FR|MR

データファイル入門[1]

ナツメ出版企画＝編

ナツメ社

まえがき

この本は書名にもあるとおり、NECのパソコン

PC−8801MKⅡFR/MR

に関するディスク操作を解説したものです。

ビジネスなど実務面でたくさんのデータを扱おうと思うと、どうしてもディスクにたよらざるを得ません。

そこで、いままでベーシックで組んでいたプログラムをディスクバージョンに変更しなければならないわけですが、ディスクでのデータの扱いは考え方を多少異にするため、すぐになじめない一面があります。そして、この理由のために、途中で挫折してしまうか、あるいはせっかくディスクを持ちながら、これをプログラムのセーブとロードのためだけに使用することにとどまっている人が多いようです。

そこで、ベーシックとディスクベーシックとの連絡をつけ、みなさんができるだけ抵抗なく入っていけるようにと考え、本書を発行しました。

ディスクは、あなたの努力次第でずいぶん実務面で活用できます。せっかくディスクがあるのですから、これを活かしてみませんか？実行画面を多くかかげましたので大変わかりやすくなっています。

なおPC−8801MKⅡFR/MRには、次の４つのモデルがあります。

- ❶ **FRモデル10（ディスク装置なし）**
- ❷ **FRモデル20（１ドライブのディスク装置内蔵）**
- ❸ **FRモデル30（２ドライブのディスク装置内蔵）**
- ❹ **MR（２ドライブのディスク装置内蔵）**

本書は、このうち❸のタイプを使って解説を進めています。❶の機種をお持ちの方は、ディスクユニットを購入し外部装置として取り付けてください。また、モデル20をご使用の方は、ドライブ２を使えませんので、本文中のファイルディスクリプタのドライブ番号指定を取り除いて読んでいただくことになります。

本書では、ベーシックの基本的なコマンド等には説明を加えておりませんので、プログラミングの入門編を終えてない方は、**「PC−8801MKⅡFR/MRプログラミング＆オペレーション入門」**（ナツメ社）を読んでから、本書にお進みください。

編者

もくじ

第 1 章

ディスクの基本操作

1･1 操作する前に

この章では、ディスクの基本的な操作方法について学びますが、本題に入る前に、いくつか注意を述べておきます。

フロッピィディスク

まず、ディスクのことを正式には**フロッピィディスク**といい、記憶容量の違いにより、**5インチ両面倍密度ミニフロッピィディスク**(FR用)、**5インチ高密度フロッピィディスク**(MR用)がありますが、**この本では単に「ディスク」または「フロッピィディスク」と呼ぶことにします。**

システムディスク

また、ディスクユニットを起動し、ディスクへのオペレーションができる状態にするためのデータが書き込まれているディスクを**システムディスク**といいます。

本体内蔵のディスク装置の場合、ディスクを1枚使用できるもの(FRのモデル20)、2枚使用できるもの(MRおよびFRのモデル30)の2種類が、また、FRモデル10の場合、使用するディスクタイプによってディスク装置も分かれますが、実質的な操作方法には違いがありません。

本書では、PC－8801MKⅡFRのモデル30を使用して説明を進めていますので、ディスクのコピーを作る操作はモデル20の場合異なってきますが、本書での解説を読み、ユーティリティプログラムを実行し画面に表示されるメッセージに従って行なえばよいでしょう。

また、FRのモデル10の場合、ディスクを起動したとき画面に表示される数字などは多少異なりますが、本書での操作には影響ありません。

要するに、本書で作るプログラムはいずれの機種でも何ら支障なく動きますのでご安心ください。

また、システムディスクにはカタカナ表示のN88－BASICと、漢字の使えるN88日本語BASICがありますが、本書ではデータをどうたくわえ、それをどう利用するか、その基本をマスターすることが目的ですので、**N88－BASIC**によって解説を進めます。

なお、本書での記事は、ナツメ社発行の「PC－8801MKⅡFR／MRプログラミング＆オペレーション入門」と重複しますが、すでに理解している方は次章へお進みください。

1·2 システムディスクのコピーを作る

システムディスクには

❶N88日本語BASIC　システムディスク
❷N88－BASIC　システムディスク

の2種類があります。

これらはいずれも、ディスクを起動させるための非常に大切なものです。何らかの手違いでこわれることもありえます。万一の場合にそなえて、このシステムディスクとまったく同じものを作っておきましょう。つまり、**システムディスクを、新しい未使用のディスクにコピーしておく**のです。

この節で出てくる命令は、まだ説明していないのでわからないかもしれませんが、書いてあるとおりにキーインしていってください。また、ここではPC－8801MKⅡFRのモデル30を対象に説明を進めますので、他の機種をご使用の方は、要領は同じですから、まずこの記事を読んでから実行してみてください。

なお、ここでは新しいディスクを4枚必要としますので、次の要領で準備してください。

❶PC－8801MKⅡFRの場合、5インチ両面倍密度版（5インチ2Dといいます）を4枚
❷PC－8801MKⅡMRの場合、5インチ高密度（5インチ2HDといいます）を4枚

〈注〉「5インチ」のディスクは正確には5.25インチですが、本書では5インチで通します。また、未使用のディスクはこれからも使いますので、できたら5枚とか10枚用意しておくとよいでしょう。

システムの立ち上げ

この操作はもうわかりますね。

ディスプレイのスイッチを入れ、次に本体の電源をONにし、オリジナルのシステムディスクをドライブ1に入れてリセットボタンを押します。

すると、アクセス表示用LEDが点灯し、使用するファイル数を問うメッセージが次のように表示されますから、「5」を入力してリターンキーを押しましょう。

How many files (0–15) ?

すると画面は次のようになりました。

これでN88－BASICが立ち上がりました。

```
Disk version [May 24,1985]
How many files(0-15)? 5
NEC N-88 BASIC Version 2.0
Copyright (C) 1981 by Microsoft
47894 Bytes free
Ok
■
```

なぜ「5」をキーインしたかといいますと、リターンキーを押すクセがついてしまいますと、ディスクで本格的にファイル操作をするようになったとき、エラーがでたりするからです。つまり、使用ファイル数を入力せずにリターンキーを押しますと、**ファイル数は自動的に2に設定されます**。しかし、ビジネス用のプログラムなどでは、ファイルを同時に3つ以上使うことはよくありますから、このような場合ファイル数が不足しエラーとなってしまいます。

ファイルを画面に表示する

それでは、次のステートメントをキーインしてみてください。キーインは大文字でも小文字でもかまいませんよ。

```
files 1 ↵
```

これは、**ドライブ1に入っているシステムディスクの中のファイルをすべて表示する命令**です。

「1」という数字がドライブ番号を示しています。ただし、ドライブ1の場合に限り、この数字を省略することができます。

もうおわかりと思いますが、**ドライブ2のディスクのファイル名を見たい場合には、パラメータに「2」を指定**して

```
files 2 ↵
```

FILESコマンド

とキーインすればよいのです。現在は、ドライブ2にはディスクは入っていませんから、これは実行しないでください。

さて、画面は次のようになりました。

```
files
copy2  .j88 6   dskut2.j88 8   sysgn2.j88 5   copy1  .j88 4   dskut1.j88 6
sysgn1.j88 5    xfiles.j88 7   copy2 *bin 1   dskut2*bin 2    sysgn2*bin 1
copy1 *bin 2    dskut1*bin 1   sysgn1*bin 1   xfiles*bin 2    @load .v1  2
@load .v2  2    @exst *v1  2   @exst *v2  2   demoex.n88 3    mek   .    2
sks   .    7    MtFuji.voi 2   train .voi 1   snrdrm.voi 1    human .voi 1
water .voi 2    steel .voi 2   clap1 .voi 1   clap2 .voi 2    tkoff1.voi 2
tkoff2.voi 2    kakkou.voi 1   cat   .voi 1   dog1  .voi 1    dog2  .voi 1
lion  .voi 1    CSMdrv*    1   kakkou*csm 1   cat   *csm 1    dog1  *csm 1
dog2  *csm 1    lion  *csm 1   @cc4h .j88 1   @cc4h *rel 1    @cc4h *map 1
@cc4h *bin 2    @cc4v .j88 1   @cc4v *rel 1   @cc4v *map 1    @cc4v *bin 2
@cc4m .j88 1    @cc4m *rel 1   @cc4m *map 1   @cc4m *bin 2    @cc2  .j88 1
@cc2  *rel 1    @cc2  *map 1   @cc2  *bin 2
Ok
■
```

ずいぶんいろいろな欧文の名前が表示されましたね。この中の１つ１つ、たとえば左上の**「copy 2 .j88」**などを**ファイル名**といいます。そして、そのうしろの数字**「6」**は、このファイルがディスク上で占有している**容量**（単位は**「クラスタ」**）を示しています。

ファイルにはそれぞれ意味がありますが、いまはこのくらいの知識でよいでしょう。

ユーティリティプログラムのロード

画面をもう一度ごらんください。一番上の行に

dskut2. J88
dskut1. J88

という２つのファイルがあります。これが、ディスクをコピーしたりする**ユーティリティプログラム**です。

ファイル名に「1」という数字が入っているほうは１ドライブのモデル20型のための、「2」が書かれているほうはモデル30型のためのユーティリティプログラムです。

ここでは、FRのモデル30を対象にしますので、後者を使います。

このプログラム（ファイル）をディスクからコンピュータのメモリ（記憶領域）に呼び出して、これを実行することによりシステムディスクのコピーを作ります。

次のようにキーインしてください。ファイル名（プログラム名）はすべて小文字で入力してください。

〈注〉コンピュータ、ディスクを管理するのに便利なプログラムのことをユーティリティプログラムといいます。

```
load "dskut2.j88"
```

- load ── f・1 キーを押す
- . ── 小数点
- " ── ダブルクォーテーション

キーインできましたら ⏎ キーを押してください。ドライブ1のランプが赤く点灯しロードが開始され、「Ok」が表示されれば終了です。

```
load "dskut2.j88"
Ok
■
```

Ok ← プログラムのロード終了

さあ、それではこのプログラムを実行しましょう。 f・1 キーを押してください。すると画面に、

```
機械語プログラムを読み込み中です。しばらくお待ちください。
```

というメッセージが表示され、少し間をおいて次のようなメニュー画面が現れます。

```
＊　＊　　ディスクユーティリティ　　＊　＊

　1　フロッピーディスクのコピー
　2　システムディスクの作成
　3　データディスクの作成
　4　IDセクタの書き換え
　5　終了

実行したい処理の番号を入力してください？　■
```

私たちは料理のメニューのことはよく知っていますね。料理がたくさん書いてあって、その中から好きなものを1つ選ぶようになってい

ます。パソコンの場合にも同じです。いろいろな処理名が画面に表示されて、その中から1つ選ぶようになっています。
ところで、私たちはシステムディスクのコピーをしようというわけですから、上のメニュー画面の1番

フロッピーディスクのコピー

を選べばよいわけです。
番号を入力するようたずねていますからキーインしてみましょう。

1 ⏎

すると画面は次のようになります。

```
[  フロッピーディスクのコピー  ]

コピーしたいフロッピーディスクをセットしたドライブ番号？ ■
```

私たちの場合、コピーしたいフロッピィディスクは「N88－BASICシステムディスク」であり、これはすでにドライブ1にセットしてありますから、1 キーを押して ⏎ です。

1 ⏎

すると今度は次のように聞いてきます。

```
新しいフロッピーディスクをセットしたドライブ番号？ ■
```

それでは、用意してある**未使用のディスクをドライブ2にセット**して（ラベルが上になるように）次のようにキーインしてください。

2 ⏎

今度は**物理フォーマット**をするかどうか聞いてきます。　物理フォーマット

```
物理フォーマットをしますか（y/n）？ ■
```

物理フォーマットとは、市販されている5インチのフロッピィデ

ィスクはそのまま使うことができないので、使えるように初期化することをいいます。これ以上の説明は省略しますが、とにかく物理フォーマットという作業をしないと使用できない、と覚えておいてください。

ここでは次のようにキーインしましょう。

y ⏎

すると画面に次のような表示が現れ、物理フォーマット実行の確認を求めてきます。

ドライブ2のディスクをフォーマットします。
よろしいですか（y/n）？ ■

やはり「y」をキーインします。

y ⏎

すると、

物理フォーマット中です。

と表示され、**ディスクの初期化（イニシャライズ）**が行なわれます。しばらくして物理フォーマットが終了すると、画面に

コピー　　ドライブ　1　→　ドライブ　2
トラック　3
トラック　4

というように、ドライブ1のディスクの情報をドライブ2のディスクへ**「トラック」**という単位ごとにコピーしていることを知らせてきます。コピーが終了すると、画面は次のようになります。

終了しました。
ディスクユーティリティの作業を続けますか（y/n）？ ■

ここで「y」をキーインすれば再び10ページのメニュー画面に戻りますが、私たちは一応終了することにして

```
n ⏎
```

とキーインします。

さあ、これでシステムディスクを新しいディスクにコピーすることができたわけですが、うまくいったかどうか確かめてみましょう。

確認の方法は**FILES（ファイルズ）コマンド**を使って、ドライブ1のディスクとドライブ2のディスクのファイル名を表示させてみればよいですね。次のようにキーインしてください。

```
files ⏎
files 2 ⏎
```

画面は下の図のようになりました（実際には1画面におさまらず上へスクロールしてしまいます）。

```
files ←―――――――――――――――――――――――――――――――――――――――――――――――――ドライブ1のディスクのファイル名を表示
copy2 .j88 6    dskut2.j88 8    sysgn2.j88 5    copy1 .j88 4    dskut1.j88 6
sysgn1.j88 5    xfiles.j88 7    copy2 *bin 1    dskut2*bin 2    sysgn2*bin 1
copy1 *bin 2    dskut1*bin 1    sysgn1*bin 1    xfiles*bin 2    @load .v1  2
@load .v2  2    @exst *v1  2    @exst *v2  2    demoex.n88 3    mek   .    2
sks   .    7    MtFuji.voi 2    train .voi 1    snrdrm.voi 1    human .voi 1
water .voi 2    steel .voi 2    clap1 .voi 1    clap2 .voi 2    tkoff1.voi 2
tkoff2.voi 2    kakkou.voi 1    cat   .voi 1    dog1  .voi 1    dog2  .voi 1
lion  .voi 1    CSMdrv*    1    kakkou*csm 1    cat   *csm 1    dog1  *csm 1
dog2  *csm 1    lion  *csm 1    @cc4h .j88 1    @cc4h *rel 1    @cc4h *map 1
@cc4h *bin 2    @cc4v .j88 1    @cc4v *rel 1    @cc4v *map 1    @cc4v *bin 2
@cc4m .j88 1    @cc4m *rel 1    @cc4m *map 1    @cc4m *bin 2    @cc2  .j88 1
@cc2  *rel 1    @cc2  *map 1    @cc2  *bin 2
Ok
files 2 ←―――――――――――――――――――――――――――――――――――――――――――――――ドライブ2のディスクのファイル名を表示
copy2 .j88 6    dskut2.j88 8    sysgn2.j88 5    copy1 .j88 4    dskut1.j88 6
sysgn1.j88 5    xfiles.j88 7    copy2 *bin 1    dskut2*bin 2    sysgn2*bin 1
copy1 *bin 2    dskut1*bin 1    sysgn1*bin 1    xfiles*bin 2    @load .v1  2
@load .v2  2    @exst *v1  2    @exst *v2  2    demoex.n88 3    mek   .    2
sks   .    7    MtFuji.voi 2    train .voi 1    snrdrm.voi 1    human .voi 1
water .voi 2    steel .voi 2    clap1 .voi 1    clap2 .voi 2    tkoff1.voi 2
tkoff2.voi 2    kakkou.voi 1    cat   .voi 1    dog1  .voi 1    dog2  .voi 1
lion  .voi 1    CSMdrv*    1    kakkou*csm 1    cat   *csm 1    dog1  *csm 1
dog2  *csm 1    lion  *csm 1    @cc4h .j88 1    @cc4h *rel 1    @cc4h *map 1
@cc4h *bin 2    @cc4v .j88 1    @cc4v *rel 1    @cc4v *map 1    @cc4v *bin 2
@cc4m .j88 1    @cc4m *rel 1    @cc4m *map 1    @cc4m *bin 2    @cc2  .j88 1
@cc2  *rel 1    @cc2  *map 1    @cc2  *bin 2
Ok
■
```

どうですか？　ドライブ1の内容も、ドライブ2の内容もまったく同じですね。つまり、コピーに成功したのです。この**コピー版のディスク**には当然**システム**もコピーされていますから、**これを使ってディスクを起動させることができる**わけです。

そこで、オリジナルのシステムディスクはどこか安全な場所に保管しておき、これからは、このコピーしたシステムディスクを使うことにします。

ジョブ用ラベルを貼る

では、ドライブ1側のシステムディスクを取り出し、これはしまっておいてください。

次に、いまコピーしたドライブ2側のディスクを取り出し、新しいディスク購入時についてくる**カラーラベル**（ジョブ用ラベル）に

N88－BASIC　システムディスク　コピー

と書いて、「2D」と書いてある四角いラベルのとなりに貼っておいてください。

システムを再び立ち上げる

ラベルを貼りましたら、このコピー版のシステムディスクをドライブ1にセットしてください。そして、本体前面左側にある**RESET**と書いてあるボタンを押してください。

どうですか？　しばらくすると「**How may files　(1－15)?**」メッセージが出てきたでしょう。こうなりましたら、5を入力してリターンキーを押してください。すると8ページ上の図のようになったと思います。

さて、それでは、もう1枚用意してある未使用のディスクに、今度はもう一方のシステムディスク「N88日本語BASICシステムディスク」をコピーしておきましょう。

まず、現在の状態から

```
load "dskut2.j88"⏎
```

とキーインしてください。

しばらくして「Ok」が表示されたら　f･5　キーを押しましょう。すると、11ページの上の図のメッセージが現れますから、ここでドライブ1のシステムディスクを取り出し、もう1枚の**「N88日本語BASICシステムディスク」**を**ドライブ1**にセットしてください。次に**新しいディスク**を**ドライブ2**にセットして、11ページ以降で説明した要

領でコピー作業を行なってください。

コピーが終了しましたら、前と同じようにコピー版ディスクに

N88日本語BASIC　システムディスク　コピー

と書いたカラーラベルを貼り、オリジナル版とともに保管しておいてください。

1·3 システムだけをコピーする

９ページの画面を見てもわかるように、システムディスクにはいろいろなファイルが保存されていますが、この中には普段必要ないものがたくさんあります。デモプログラムなどでずいぶんディスクのメモリを浪費しています。ですからここでは、ディスクを立ち上げるシステム（システムはファイルではありませんよ。ディスク装置を起動させるための情報〈データ〉として、ディスクの所定の位置に書き込まれています）だけを新しいディスクにコピーすることにしましょう。

ユーティリティプログラムはまだメモリにありますから、これを、f・5 キーを押して実行しましょう。

それでは f・5 キーを押してください。すると、10ページのようなメニュー画面が表示されますから、この中から２番の

システムディスクの作成

を選択します。では、「**2** ⏎」とキーインしてください。

すると画面に

```
【 システムディスクの作成 】
システムディスクをセットしたドライブ番号？ ■
```

というメッセージが表示され、**「システムディスク」**をセットしたドライブ番号をたずねています。私たちの場合すでにドライブ１にセットしてありますから「**1** ⏎」とキーインします。

すると今度は次のように、**「新しいフロッピィディスク」**をセットしたドライブ番号をたずねてきます。

```
新しいフロッピーディスクをセットしたドライブ番号？ ■
```

では、未使用の新しいディスクをドライブ２にセットして「**2** ⏎」とキーインしましょう。

今度は、

```
物理フォーマットをしますか（y/n）？ ■
```

と、物理フォーマットをするかどうかたずねてきます。未使用のディスクの場合物理フォーマットは必要ですので「**y** ⏎」とキーインし

ます。
　すると、物理フォーマットをしてよいかどうかの確認メッセージが表示されます。

```
ドライブ２のディスクをフォーマットします。
よろしいですか（y/n）？ ■
```

　ここでは「**y** ⏎」とキーインしましょう。すると、画面に「**物理フォーマット中です**」のメッセージが表示され、続いて「**システムフォーマット中です**」と表示されます。
　システムフォーマットが終ると、「システムをコピー中です」の表示が画面に現れ、ドライブ１側のディスクの**システムプログラム**がドライブ２側のディスクに転送されます。
　しばらくして画面は次のようになって「システムディスクの作成」を終了します。

```
システムをコピー中です。

終了しました。
ディスクユーティリティの作業を続けますか（y/n）？ ■
```

　私たちは一応これで打ち切ることにして「**n** ⏎」とキーインします。
「Ok」が表示されたらドライブ２側のディスクを取り出し、カラーラベルに

　ディスクＡ：プログラム用

と書いて貼っておいてください。この**ディスクＡ**はあとで使いますから、それまで保管しておいてください。

　現在、ドライブ１にはシステムディスクが、そしてドライブ２にはセットされているディスクはありません。このままの状態で次の節へお進みください。

1・4　データディスクを作る

私たちは、あらかじめ用意した4枚の新しいディスクのうちすでに3枚を使い、残りはあと1枚となりました。

この1枚のディスクを今度はデータディスクにしてみます。

前節で作ったシステムディスクには「システムプログラム」が保存されていますから、その分だけ私たちが使える容量が少なくなっています。そこで今度は、1枚のディスクすべてを、住所録などの**データ保存用**として私たちが使えるようにしようというわけです。

まだユーティリティプログラムがメモリにありますから f・5 キーを押して実行してください。すると10ページのメニュー画面が表示されます。今度は3番目の**「データディスクの作成」**を選びますので、「**3** ↵」とキーインします。

すると画面は次のようになります。

```
【　データディスクの作成　】
新しいフロッピーディスクをセットしたドライブ番号？　■
```

未使用の新しいディスクをドライブ2にセットして「**2** ↵」としてください。

すると前節と同じように「物理フォーマット」をするかどうか聞いてきますから、物理フォーマットをすることにして前節と同じように操作してください。

これ以降の画面メッセージは「システムをコピー中です」の表示が出ないことを除けば、前ページの図と同じです。

終了したら、ドライブ2側のディスクを取り出し、カラーラベルに

ディスクB：データ用

と書いて貼っておいてください。以後、こちらのディスクを**ディスクB**と呼ぶことにします。

それではリセットボタンを押してシステムをもう一度立ち上げなおして次の節へ進みましょう。

1・5　プログラムのセーブとロード

プログラムをせっかくキーインしても、電源を切ると、現在コンピュータの記憶領域（メモリ）にあるプログラムはすべて消えてしまいます。

ですから、これをどこかにたくわえておかなければなりません。その、たくわえる場所が小さな磁気盤、ディスクです。

この節では、サンプルとして小さなプログラムを1つ作り、これをディスクに保存し、それをもう一度メモリ上に呼びだす方法を学びましょう。

すでにごぞんじの方もいると思いますが、そのような方はここを飛ばして次へお進みください。

プログラムをディスクに保存することを**セーブする**といい、**SAVE（セーブ）コマンド**を使います。

ここでは、プログラムをセーブし、それをもう一度メモリ上に取り出すことを確認すればよいので、サンプルプログラムは、ごく簡単なものにします。次のようにキーインしてください。

```
1 x=5
2 y=3
3 print x+y
```

変数XおよびYに、それぞれ5と3を代入し、その合計を画面に表示させるものです。

では、ホームクリアキーで画面をクリアし、リストコマンドで、プログラムがメモリに記憶されたかどうか確認しておきましょう。

```
list
1 X=5
2 Y=3
3 PRINT X+Y
Ok
■
```

プログラムをディスクにセーブする際、どんなプログラム名で保存しておくのか、プログラムに名前をつけなければなりません。この名前のことを**ファイル名**といいますが、ここでは仮りに「**prog**」としておきます。

次のようにキーインします。まだ [↵] キーは押さないでください。

[S][A][V][E][スペース] ["][1][:][P][R][O][G]["]

（[SAVE][スペース]["] の部分は）[f・6] キーを押すのと同じです

「save ″」は [f・6] キーに定義されていますから、これを使えば楽ですね。

また、「prog」は、ここでは小文字にしてください。

画面は次のようになっていますか？

```
save "1:prog"
```

ファイル名（prog）
コロン（:）
ドライブ番号（1）

「save」は大文字でも小文字でもかまいませんよ。ただし、「prog」だけは小文字にしてください。**ファイル名の場合、コンピュータは大文字と小文字を区別します。**

さて、私たちのコンピュータには、ディスクを入れる場所が2つついています。これを**ドライブ**といい、右側のほうを**ドライブ1**、左側を**ドライブ2**といいます。

ドライブ

上の画面ではドライブ1を指定しました。つまり、「prog」をドライブ1側のディスクにセーブすることになります。

このように、SAVEコマンドの書式は、

SAVE“〈ドライブ番号〉：〈ファイル名〉”

で表わされ、ドライブ番号とファイル名をコロンで結んだものを**ファイルディスクリプタ**といいます。そして、**ドライブ1にセーブする場合には、ドライブ番号を省略できます。**つまり上の画面では、ファイルディスクリプタを、「1：」をはぶいて「prog」としてもいいのです。

ファイル
ディスプリプタ

〈注〉 F・Rのモデル20にはドライブ2がついていませんから、「2：」は使うことができません。つまり、ドライブ番号はまったく不要になります。

それでは、いまディスクは入っていませんが、ドライブ2にセーブしたいときにはどうしたらよいでしょうか？　もうわかりますね。次のようにキーインすればよいのですね。これはキーインしないでくださいよ。**(注)**

```
save "2:prog"
```

ドライブ番号（2）

さあ、おまたせしました。リターンキーを押してみましょう。

すると、赤いランプがつき、少したってから画面にセーブ終了の合図「Ok」が表示されます。

```
save "1:prog"
Ok
■
```

それでは、ほんとうにセーブされたかどうか確認してみたいと思います。「**files** ⏎」とキーインしてください。

すると画面は次のようになりました。

```
files
copy2 .j88 6    dskut2.j88 8    sysgn2.j88 5    copy1 .j88 4    dskut1.j88 6
sysgn1.j88 5    xfiles.j88 7    copy2 *bin 1    dskut2*bin 2    sysgn2*bin 1
copy1 *bin 2    dskut1*bin 1    sysgn1*bin 1    xfiles*bin 2    @load .v1  2
@load .v2  2    @exst *v1  2    @exst *v2  2    demoex.n88 3    mek   .    2
sks   .    7    MtFuji.voi 2    train .voi 1    snrdrm.voi 1    human .voi 1
water .voi 2    steel .voi 2    clap1 .voi 1    clap2 .voi 2    tkoff1.voi 2
tkoff2.voi 2    kakkou.voi 1    cat   .voi 1    dog1  .voi 1    dog2  .voi 1
lion  .voi 1    CSMdrv*    1    kakkou*csm 1    cat   *csm 1    dog1  *csm 1
dog2  *csm 1    lion  *csm 1    @cc4h .j88 1    @cc4h *rel 1    @cc4h *map 1
@cc4h *bin 2    @cc4v .j88 1    @cc4v *rel 1    @cc4v *map 1    @cc4v *bin 2
@cc4m .j88 1    @cc4m *rel 1    @cc4m *map 1    @cc4m *bin 2    @cc2  .j88 1
@cc2  *rel 1    @cc2  *map 1    @cc2  *bin 2    prog  .    1
Ok
■
```

セーブされている（prog）

いかがですか？　いま私たちがセーブしたプログラムが見えましたね。どうやら、うまくセーブできたようです。

アスキー形式のセーブ

これで、プログラムのセーブ方法を理解していただけたと思いますが、セーブにはもう1つの方法がありますので覚えておきましょう。

いま私たちが学んだセーブ方法は**「非アスキー形式のセーブ」**といい、これから説明するのを**「アスキー形式のセーブ」**といいます。

たとえば「prog」をアスキー形式でセーブするときは、次のようにキーインします。

```
save "1:prog",a
```

a ← アスキー形式のセーブ

前回のセーブ方法と異なるのは「,a」がついている点ですね。たったこれだけでよいのです。このように「,a」をつけることから、このセーブ方法を**「Aオプションつきのセーブ」**ともいいます。

なお、「a」は大文字でもかまいません。

アスキー形式と非アスキー形式の違いはディスクへの保存のされ方が違うのですが、いまは、前者のほうが後者より多少メモリを余計必要とする、くらいに考えておいてください。

ただ、アスキー形式のほうがディスク上のメモリを多少必要とするかわりにメリットもありますが、本書では非アスキー形式のセーブで話を進めます。

プログラムのロード

LOAD

フロッピィディスクにセーブされたプログラムをもう一度コンピュータのメモリに呼びだすことを**ロードする**といい、**LOAD(ロード)コマンド**を使います。

その書式は、アスキー形式、非アスキー形式、いずれでセーブされたプログラムについても同じで、次のとおりです。

```
load "1:prog"
```

- "1:prog" — ファイルディスクプタ
- load " — ファンクションキー1に定義されています

つまり、セーブする場合の「SAVE」を「LOAD」に変えただけです。ファイルディスクリプタの表わし方も同じですね。**ドライブ1にセーブしたプログラムはドライブ1から、ドライブ2にセーブしたプログラムはドライブ2からロードします。**

それでは「prog」をさっそくロードしてみようと思いますが、本当にロードできたかどうか確認する意味で、コンピュータのメモリをクリアしてしまいましょう。

NEW

本体前面左のリセットボタンを押してコールドスタートしてもよいのですが、ここでは新しい命令を1つ覚えましょう。それは、**NEW(ニュー)コマンド**です。新しくする、新規にする、ということですね。

次のようにキーインしてください。

```
new ⏎
```

これで、メモリ内のプログラムはすべて消えてしまいました。ちな

みに、 f・4 キーを押してプログラムリストを表示させてみてください。

```
new
Ok
list
Ok
■
```

どうですか？プログラムリストは何も表示されませんね。つまり、メモリはすっかりクリアされてしまったのです。

それでは、前ページの書式にしたがってロードしてみてください。そして f・4 キーでプログラムリストを表示させてください。

画面は次のようになりましたか？

```
load "1:prog"
Ok
list
1 X=5
2 Y=3
3 PRINT X+Y
Ok
■
```

ドライブ1のディスクからロードされたプログラム（1〜3行）

成功ですね。ではプログラム名を大文字にしてためしてみましょう。

```
load "1:PROG"
File not found
Ok
■
```

エラーメッセージ（File not found）

「ファイルが見つかりません」というエラーメッセージが返ってきました。ファイル名は、小文字でセーブしたら、小文字でロードしなければいけないのです。

1·6 ファイルの転送

ファイルズコマンドによりディスクの内容を画面に表示させると

dskut2. j88
dskut2＊bin

などのようなファイル名を見ることができます。

このうち、1番目の名前のように**ピリオド**が入っているものはセーブ、ロードできますが、2つ目の名前のように**アスタリスク**が入っているものは機械語プログラムですからLOAD、SAVEコマンドを使用できません。そこで登場するのが「ファイルの転送」です。

では始めますが、**N88－BASICシステムディスク**を**ドライブ1**に、1・3節で作った**ディスクA**を**ドライブ2**にセットしてください。

セットできたら次のようにキーインしてください。

```
load "xfiles.j88" ⏎
```

ドライブ1側のシステムディスクに入っている「xfiles.j88」という名前のファイルが転送のためのプログラムです。

「Ok」が表示されたら f・5 キーで実行してください。

すると「機械語プログラムを読み込み中です。しばらくお待ちください。」のメッセージに続いて次のように表示されます。

```
《 ファイル 転送 》
マスターディスクのドライブ番号? ■
```

マスターディスクというのは、転送したいファイルが入っているディスクのことで、私たちの場合、ドライブ1側のシステムディスクがこれに相当しますから「**1** ⏎」とキーインします。

すると

```
転送先ディスクのドライブ番号? ■
```

と、どのドライブのディスクに転送するのか聞いてきます。私たちはドライブ2のディスクAに転送しますので「**2** ⏎ とキーインします。

今度は画面は次のようになり、ディスク媒体の確認です。

```
5.25インチフロッピーディスクのディスクタイプを指定してください。
  1. 両面倍密度（2D）
  2. 片面倍密度（1D）

ドライブのディスクタイプ？ ■
```

内蔵タイプのディスクの場合「2D」（これはFRの例です。MRの場合「2HD」です)ですから「**1** ⏎」とします。

次に同様に「**ドライブ2のディスクタイプ？**」と聞いてきますから、やはり「**1** ⏎」とキーインします。

すると次のように、これまでの入力で間違いないかどうか確認してきます。

```
《 ファイル 転送 》

マスターディスク・・・・ドライブ1 （5.25インチ両面倍密度）
転送先ディスク・・・・・ドライブ2 （5.25インチ両面倍密度）

ドライブ1からドライブ2へファイルを転送します。
よろしいですか（y/n）？ ■
```

間違いなければ「y」、ダメなら「n」をキーインします。「n」をキーインすると、もう一度やりなおしです。ここでは「y」を入力してみましょう。「**y** ⏎」です。

すると今度は

```
すべてのファイルを転送しますか（y/n）？ ■
```

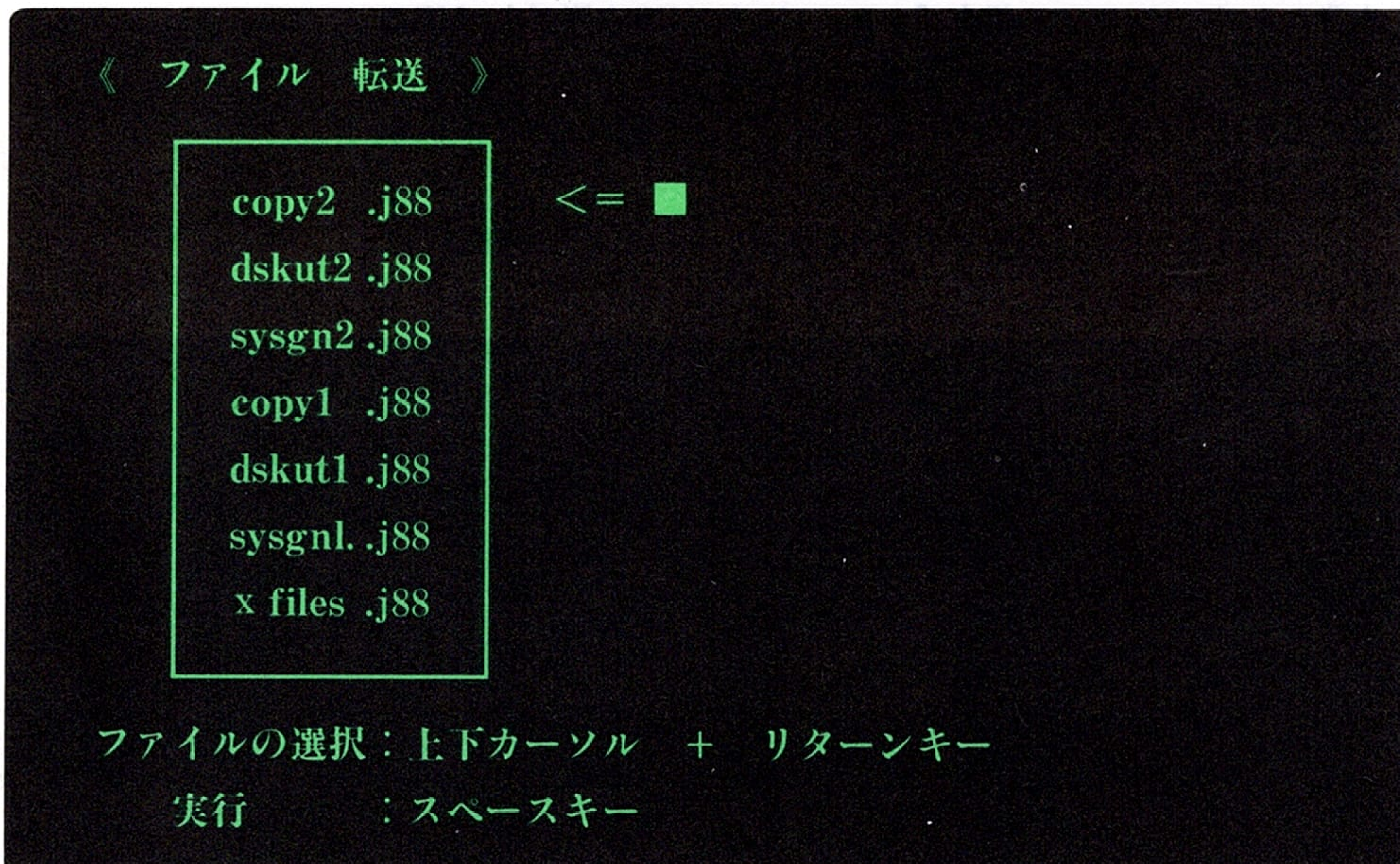
とたずねてきます。

ドライブ1の「N88－BASICシステムディスク」には、いろいろなファイルがたくわえられています。これらすべてのファイルをドライブ2側に転送しますか、ということです。

私たちは、ディスクユーティリティプログラム**dskut2. J88**と対になっている**dskut2＊bin**だけを転送することにしますので、ノー、つまり「**n** ⏎」と入力します。

1・2節では「dskut 2 . j88」を実行してシステムディスクのコピーを作りましたが、そのとき「機械語プログラムを読み込み中です」の表示がでましたね。そのときに実は、機械語プログラム「dskut 2 ＊bin」を読み込んでいたのです。つまり、ディスクユーティリティプログラムはこの2つが組になって始めて動作するのです。

さて、画面は次のようになっています。



現在、矢印が「copy 2 . n88」のところを指し、そのうしろでカーソルが点滅しています。

キーオペレーションは、下のメッセージに書いてあるように ↑ ↓ キーを使って矢印「⇐」を上下に移動することができます。

つまり、転送したいファイルがあったらカーソル移動キーで矢印をそのファイル名のところへ合わせるのに使います。もし、その矢印が指しているファイルに間違いなければリターンキーを押します。

ちょっとリターンキーを押してみてください。どうですか？ 矢印が指しているファイル名が白黒反転表示になりましたね。これが転送すべきファイルを**「選択」**した状態です。このファイルの転送を実行したいときはスペースキーを、反対に、これを解除したいときはもう一度リターンキーを押します。

もう一度リターンキーを押してみてください。ファイル名がノーマ

ルな状態で表示されましたね。

さて、私たちが転送するファイル名を探しましょう。[↓]キーを使って矢印を一番下へもっていってください。一番下へ行ったら、さらに[↓]を押してみてください。１回押すたびに１つずつスクロールアップし、新しいファイル名が一番下に現れます。

２回押したところで探している「**dskut2＊bin**」が現れましたね。ではこれを選択することにしてリターンキーを押します。するとファイル名が白黒反転表示されますから、今度はスペースキーを押して実行しましょう。スペースキーを押してみてください。

すると「**よろしいですか（y/n）?**」と確認してきます。「**y** [⏎]」とキーインすると画面に

```
ドライブ1からドライブ2へ転送中です。

          dskut2＊bin    1  →  2
```

の表示が出てファイルの転送が実行されます。終了すると、「ファイル転送ユーティリティの作業を続けますか（y/n）?」と聞いてきますから、これで終ることにして「**n** [⏎]」とキーインします。

以上でファイルの転送は終わりです。

それではここで、ドライブ２側の**ディスクA**にディスクユーティリティプログラム**dskut2.j88**をセーブしておきましょう。次のようにまずプログラムをロードします。

```
load "dskut2.j88"⏎
```

「Ok」が出たら、これをドライブ２側にセーブします。

```
save "2:dskut2.j88" ⏎
```

「Ok」が現れたらFILESコマンドで確認しておきましょう。

```
files 2
dskut2*bin 2  dskut2.j88 8
Ok
■
```

うまくいっていましたね。

1・7　ファイルディスクリプタについて

ここではファイル名のつけ方について、簡単に説明しておきましょう。

”ドライブ番号：ファイル名．拡張子”

- 3文字以内(つけなくてもかまいません) — 拡張子
- 拡張子をつける場合の区切りとなる小数点 — ．
- 1文字以上6文字以内 — ファイル名
- コロン — ：

ドライブ番号

ドライブ番号は前に説明したように、ドライブ1の場合には省略することができますが、ドライブ2の場合には必ずドライブ番号を入れ、コロンで区切ります。

ファイル名

ファイル名は6文字以内で、英字、数字、カナ文字など、いずれも使用することができます。ただし、英文の大文字、小文字は別のものとして扱われますので注意してください。

次の**小数点**は、拡張子をつける場合に区切りとして使います。

拡張子

最後の**拡張子**、これはなくてもかまいませんが、あとになってファイルを管理する場合大変役立ちます。

なお、ファイルと拡張子を合わせたものをファイル名と考えてもよいのですが、ファイル管理上、区別しておいたほうが便利ですので、本書では分けて考えます。

拡張子のメリットについてですが、たとえば「prog」を例にとりますと、次のようにしておくと大変便利です。

prog.bas	ベーシックによるプログラムファイル
prog.dat	データファイル
prog.asc	アスキーセーブしたプログラムファイル
prog.asm	マシン語プログラムファイル

「bas」は basic の略、「dat」は data の略、「asc」は ascii の略、「asm」は assemble の略です。

なお、上の形でディスクにセーブして FILES コマンドで画面に表示した場合、データファイルおよびアスキー形式のプログラムファイルには小数点は入りません。また、マシン語プログラムの場合には、「*」が小数点のかわりに表示されます。

1·8　ファイル名の変更と削除

何かのつごうでファイル名を変更したり削除したりしたい場合が生ずると思います。ここでは、その方法について簡単にふれておきます。

ファイル名の変更

ディスクに書き込まれているファイル名の変更は、次の書式に従って行ないます。**NAME（ネーム）コマンド**を使います。

NAME 〈旧ファイルディスクリプタ〉 AS 〈新ファイルディスクリプタ〉

ファイルディスクリピタ中のドライブ番号は、新旧いずれも同じでなければいけません。

たとえば「prog」を「PROG」のように大文字に変更したい場合、次のようにキーインします。ディスクは仮りにドライブ1に入っているとしましょう。

```
name "prog" as "PROG"
```

これでファイル名が変更されます。ためしてみてください。そしてFILESコマンドで確認してください。

ファイル名の削除

プログラミングの過程で、サンプルプログラムとかテストプログラムなど、いろいろなプログラムがディスクにセーブされていきます。そんなとき、すでに不要になったものは、次の**KILL（キル）コマンド**でディスクから削除しておきましょう。

書式は次のとおりです。

KILL 〈ファイルディスクリプタ〉

たとえば、ドライブ1のディスクに入っている「prog」を削除したい場合、次のようにキーインします。

```
kill "prog"
```

ドライブ1ですから、ドライブ番号は省略してもかまいません。

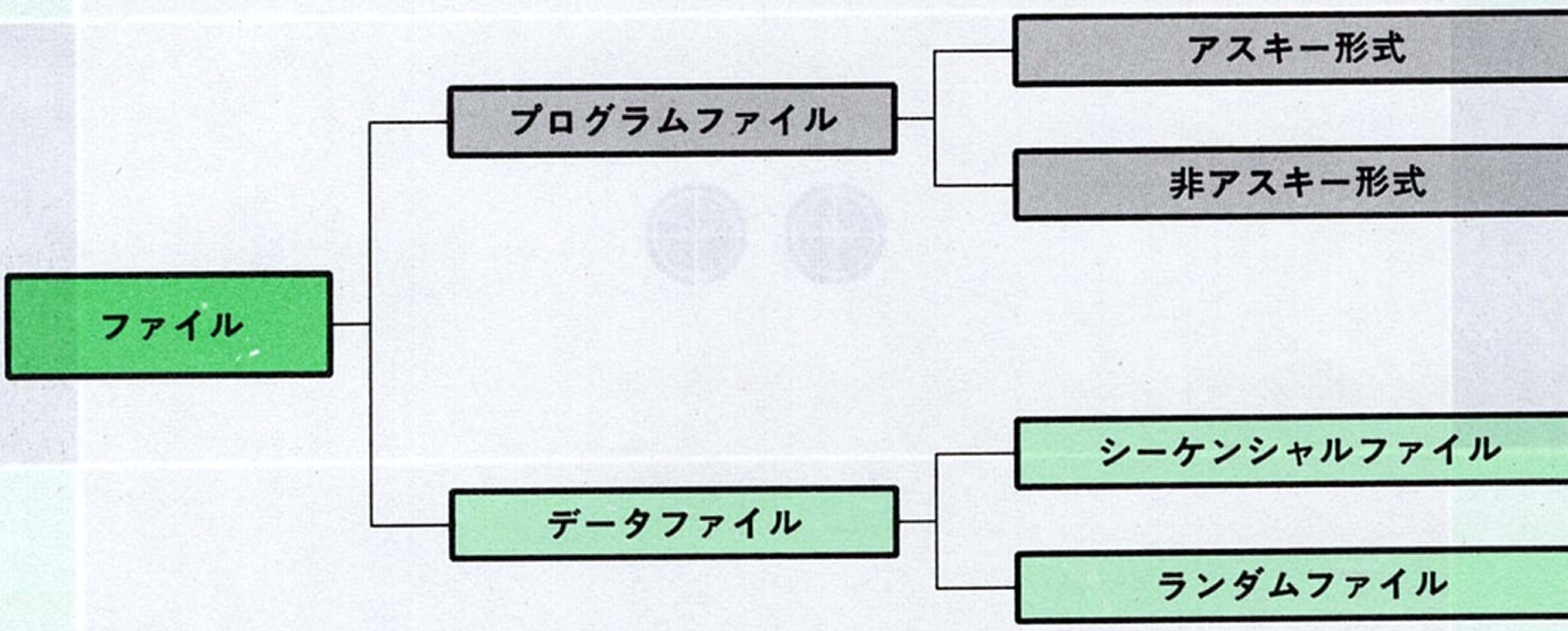
間違えて、大切なプログラムを削除しないよう、くれぐれも注意してください。一度KILLすると復帰できませんので……。

第2章

シーケンシャルファイル[I]

2・1　ファイルの種類

情報の集合体のことをファイルといいますが、これはその性質によって次のようにわけることができます。



この章で学習するのは**データファイル**のうちの**シーケンシャルファイル**のほうですが、違いをはっきりさせるために、**ランダムファイル**についても簡単にふれておきます。

その前に、プログラムファイルとデータファイルの違いはもうおわかりですね。

プログラムファイル
データファイル

そうです、**BASICプログラムなどをフロッピィディスクにセーブした場合にプログラムファイル、電話番号・住所・氏名といったデータをフロッピィディスクに書き込んだ場合にその集まりをデータファイルといいます。**

シーケンシャルファイルもランダムファイルも同じデータファイルなのですが、フロッピィディスクに書き込まれる状態が違います。

「シーケンシャル」というのは「連続的な」という意味です。「ランダム」というのは直訳すれば「デタラメナ」ということですが、デタラメというよりは「連続的ではない」くらいに理解しておいたほうがよいようです。

シーケンシャルファイルとは

シーケンシャルファイルとは、ディスクに1番目のデータから連続的に書き込まれたファイルのことで、これをディスクから読み出そうとするときに、**書き込まれた順序でしか読み出せません。**

たとえば電話帳のファイルをシーケンシャルで作った場合、夏目さ

んという人の電話番号を知りたくても、もし50音順に書き込まれていれば、安藤さん、伊藤さん…と、夏目さんの前に登録してある人のデータを読み出していかなければなりません。

あなたは、カセットテープを使ってファイルを作りましたか？ あれがシーケンシャルです。

シーケンシャルファイルの特徴をまとめると次のとおりです。

シーケンシャルファイルの特徴

❶ データをロード・セーブする場合、必ずファイルの先頭から順次行います。

❷ ファイル内の一部だけを書き換えることができません。

❸ ファイルを変更できるのは、ファイルの最後に新しいデータを追加することだけです。

❹ プログラム面、あるいはファイル設計の面から見ても、ランダムファイルに比較して簡単です。

❺ 先ほどの電話の例でいいますと、夏目さんの電話番号・住所・氏名を１レコード（このレコードの集まりがファイルです）と呼びますが、この大きさは255文字以内でしたら自由に変えられます。

レコード

意味があまりわからない人もいるかもしれませんが、「ランダムファイルより扱いが簡単だ」くらいに考えていただき読み進んでください。あとで実際にプログラムを組んでみればわかると思いますし、そのときになってもう一度読みなおしていただいてもけっこうです。

ランダムファイルとは

次にランダムファイルですが、やはり電話帳の例にそってお話しします。

電話帳ファイルをランダムファイルとしてフロッピィディスク上に作った場合、たとえば夏目さんにかかわるデータを知りたいとき、ほかの関係ないデータまで読み出す必要はなく、直接に取り出すことができます。したがって、時間的なロスもなく大変効率的です。

ランダムファイルについても特徴を列挙しておきましょう。

ランダムファイルの特徴

❶ ファイル内の各レコードを任意に読み書きできますので、処理時間を短縮できます。

❷ ファイル内の一部のデータだけを書き換えることができます。

❸ データの追加も自由です。

❹ シーケンシャルファイルに比べてプログラムが複雑になります。

❺ １つ１つのレコードの大きさは256バイト以内の一定バイト数に固定されています。

2・2　ディスクの構造

この節ではPC－8801MKⅡFR/MRの内蔵ディスクで使用できるディスクについて説明しますが、まだ実際にプログラムによるファイル操作をしていませんので少しわかりにくいかもしれません。とくに、ディスクの構造がどうなっているかということは、プログラミングする上であまり大切ではありません。ですから、ちょっととまどいを感じる人は、この節をとばして次へ進んでいただいてもかまいません。

FR/MRで使えるディスクは次の２つです。

❶　5インチ高密度フロッピィディスク（MR用）

❷　5インチ両面倍密度フロッピィディスク（FR用）

呼び名は多少違いますが、扱い方はどれも同じです。違うのは記憶容量が多いか少ないかということになります。

右ページをごらんください。構造を図解してあります。

セクタ

いずれも**「セクタ」「トラック」「サーフェス」**という単位で構成されていて、**読み書きはセクタ単位で行われます。**

1セクタは256バイト、文字に換算すると256字分ということになります。このセクタが図の円の1周分、5インチ高密度ですと26セクタ、5インチ両面倍密度のディスクですと16セクタ集まったものを**トラック**と呼びます。**サーフェス**とは裏か表かということです。

トラック

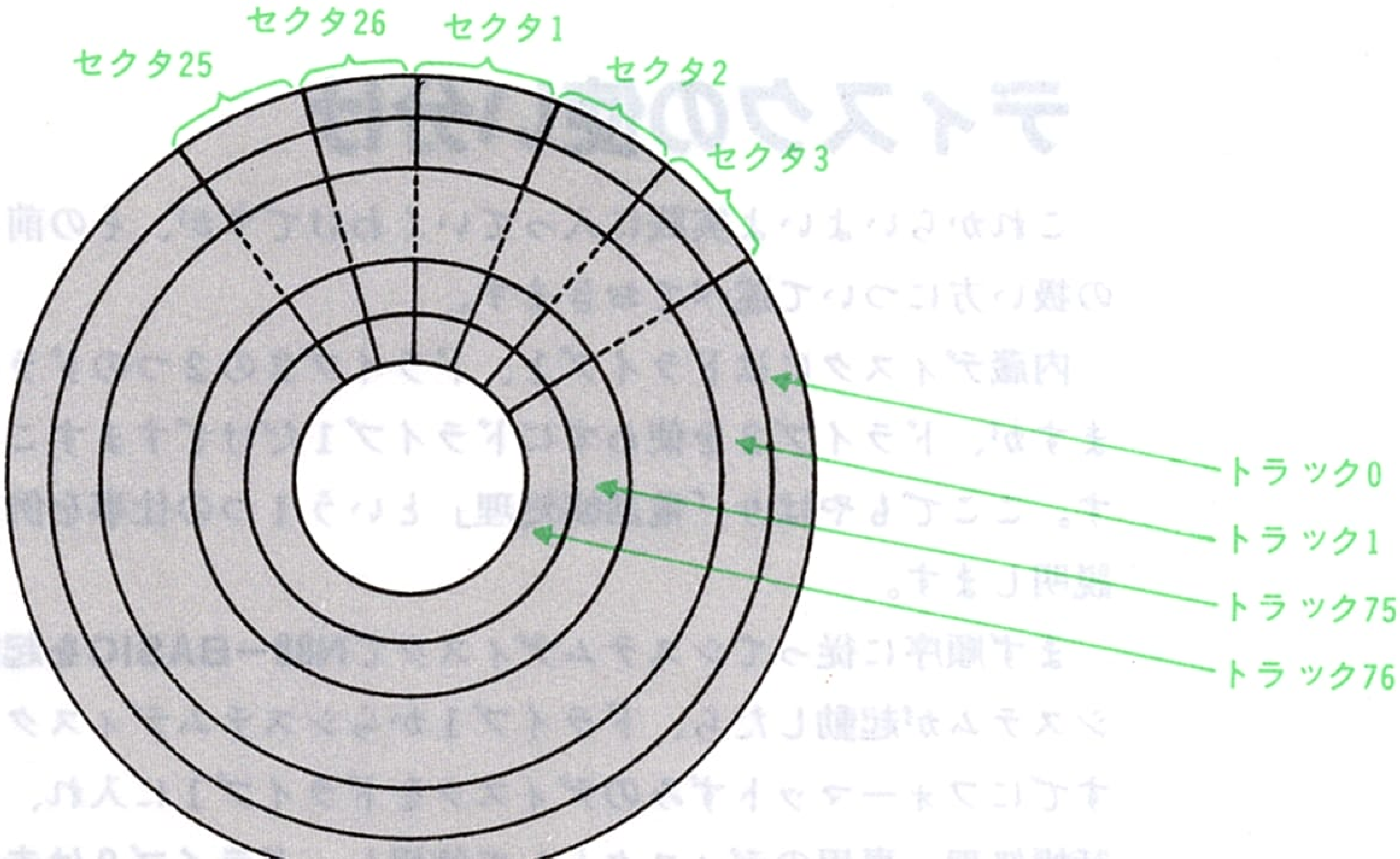
サーフェス

ディスクの種類によって総セクタ数が違ってきます。これらを表にまとめておきます。

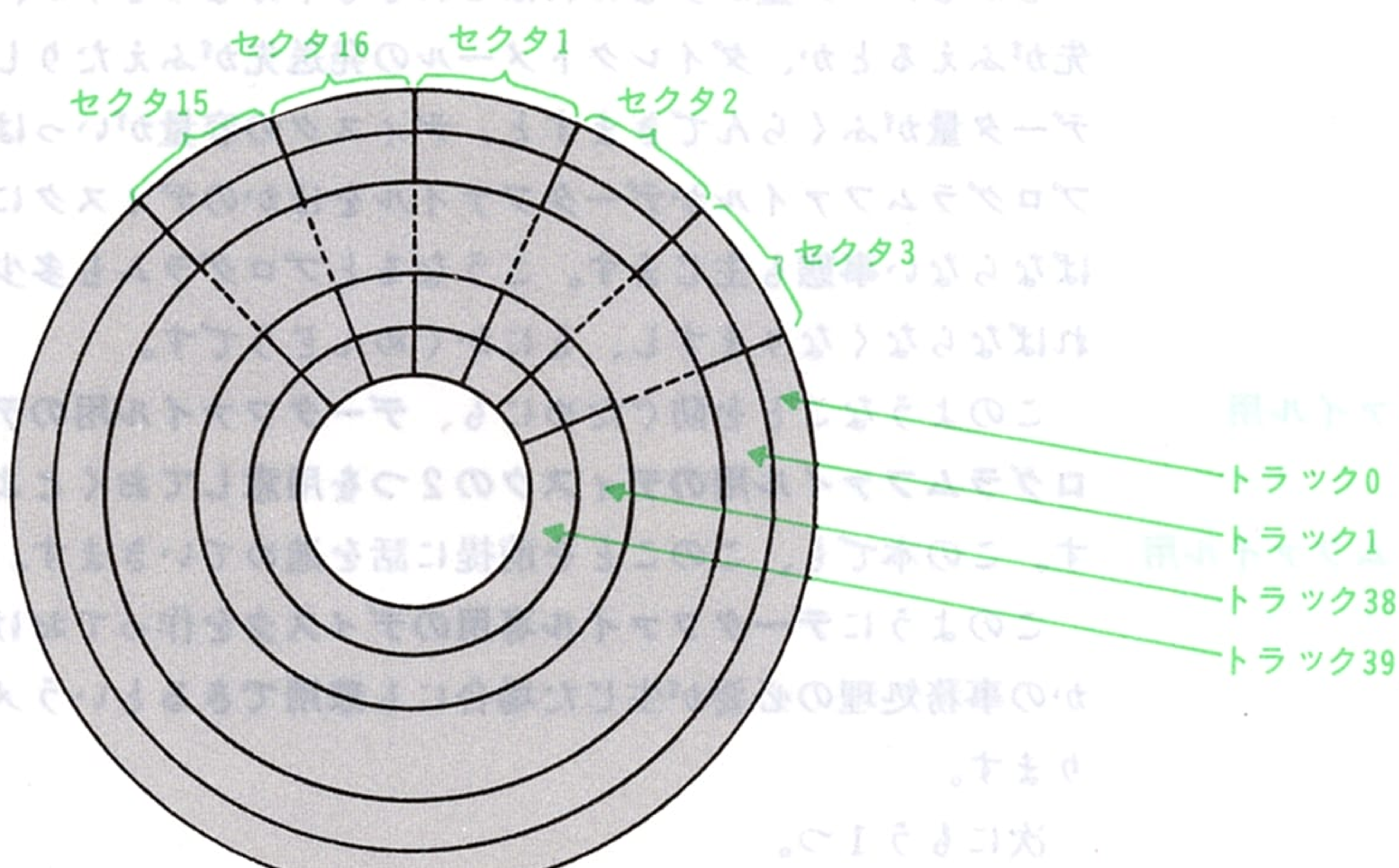
また、総セクタ数は次のように計算します。

総セクタ数＝1トラック当たりのセクタ数×トラック数×サーフェス

フロッピィディスク	サーフェス	トラック数	セクタ数	総セクタ数
5インチ高密度	0か1か	77	26	4004
5インチ両面倍密度	0か1か	40	16	1280



●5インチ高密度フロッピィディスク



●5インチ両面倍密度フロッピィディスク

2・3　ディスクの扱い方

ディスクの使い分け

これからいよいよ実践に入っていくわけですが、その前にディスクの扱い方について述べておきます。

内蔵ディスクにはドライブ1、ドライブ2の2つのドライブがありますが、ドライブ2を使わずにドライブ1だけですますこともできます。ここでもやはり「電話帳処理」という1つの仕事を例にとってご説明します。

まず順序に従ってシステムディスクで**N88−BASICを起動**させます。システムが起動したら、ドライブ1からシステムディスクを取り出し、すでにフォーマットずみのディスクをドライブ1に入れ、これを「電話帳処理」専用のディスクとして使用し、**ドライブ2は未使用**のままにしておくというやり方です。

この場合、いろいろな処理をするためのプログラムも、電話番号とか氏名といったデータも、1つのディスクに押し込んでおかなければなりません。

しかしデータ量が少なければこれでも十分なのですが、やがて得意先がふえるとか、ダイレクトメールの発送先がふえたりして保存するデータ量がふくらんできますと、ディスクの容量がいっぱいになり、プログラムファイルかデータファイルをほかのディスクに移さなければならない事態も生じます。こうなるとプログラムも多少変更しなければならなくなりますし、とにかくめんどうです。

データファイル用ディスク

プログラムファイル用ディスク

このようなことを防ぐためにも、**データファイル用のディスクとプログラムファイル用のディスクの2つを用意しておくとよい**と思います。この本でも、このことを前提に話を進めていきます。

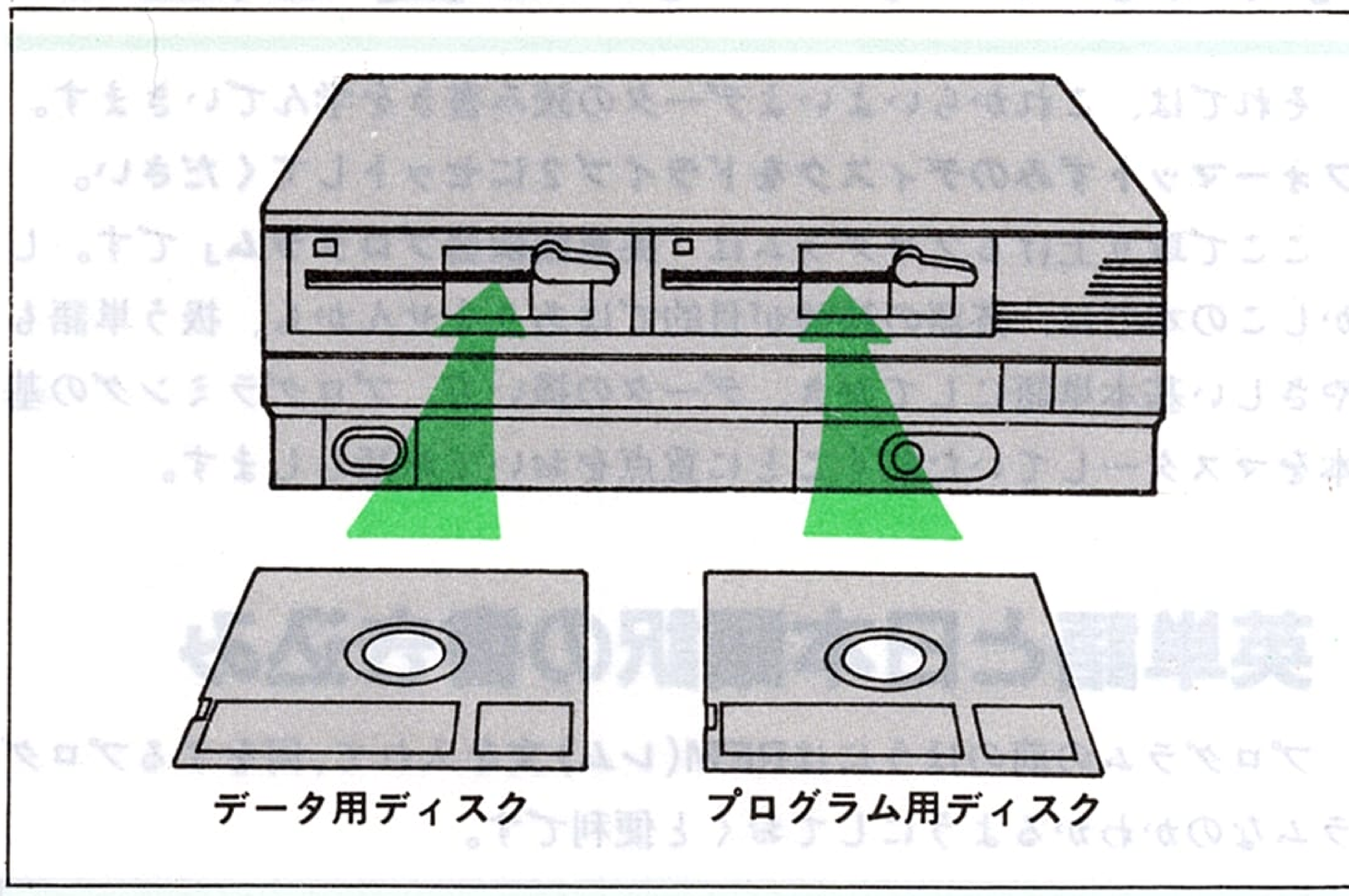
このように**データファイル専用のディスク**を作っておけば、何かほかの事務処理の必要が生じた場合にも**兼用できる**というメリットもあります。

次にもう1つ。

データファイル用のディスクはドライブ2に入れ、プログラムファイル用のディスクはドライブ1に入れる習慣をつけるとよいと思います。これも、あなたがコンピュータによる事務管理に上達し、扱うデータ量が多くなったときにおわかりいただけると思います。

というようなわけで、本書では次節から**ディスクA**(17ページ参照)

をドライブ1に、**ディスクB**（18ページ参照）をドライブ2にセットしてプログラミングの学習をすることにします。



ラベルを貼っておく

ディスクが1枚2枚のうちはいいですが、ふえるにつれて、何のためのディスクかはっきりさせておく必要があります。未使用のディスク購入時についてくる**ジョブ用ラベル**をディスクに貼り、「給与計算」「在庫管理」などの仕事名を書き込んでおくとよいでしょう。また、**すでに貼ってあるラベルに書くときはディスクにキズがつかないようにサインペンなどを使ってください。**

ジョブ用ラベル

コピーをとっておく

この作業も忘れてはならないことの1つです。

せっかく苦心して作り上げたデータファイル、あるいはプログラムファイルの入ったディスクをどんなことで壊してしまうかしれません。少しめんどうでも、プログラムを修正したり、データを追加したりしたときにはユーティリティプログラム「**dskut 2**」を走らせメニューから**「フロッピーディスクのコピー」**を選択して複写を行ってください。コピーしたディスクは大切に安全な場所に保管しておき、変更のつど取り出してコピーしなおすようにしましょう。また、2回目以降のコピーのときは「物理フォーマット」は必要ありません。

2・4　ディスクにデータを書き込む

それでは、これからいよいよデータの読み書きを学んでいきます。**フォーマットずみのディスクをドライブ2にセットしてください。**

ここで取り上げるプログラムは**「英単語練習プログラム」**です。しかしこの本では、英語の勉強が目的ではありませんから、扱う単語もやさしい基本単語にしておき、データの扱い方、プログラミングの基本をマスターしていただくことに重点をおいてお話しします。

英単語と日本語訳の書き込み

〈注〉REM文は「リマーク文」とも読みます。

プログラムの前のほうには**REM(レム)文**を入れて、何をするプログラムなのかわかるようにしておくと便利です。

```
100 ' file name ････> eitan1.bas   ←ファイル名
110 ' 1985.2.15                    ←作成・修正年月日
120 ' シーケンシャル データ ノ カキコミ ←処理内容
130 ' save "eitan1.bas"            ←プログラムのセーブ用
140 '
```

プログラムのセーブ法

行番号130のステートメントは、**プログラムを手直ししたのちセーブするとき便利**です。

つまり、カーソルを「save」のsの上に重ね、DELキーを6回たたくとsが行頭に移動しますから、ここでリターンキーを押せば、このプログラムをセーブすることができます。

まず、画面の桁数と行数ですが、小さい文字のほうがしまった感じがしますので、**80桁×20行**にします。また、**ファンクションキー表示も不要**ですので消しておきます。

```
                                  ┌ファンクションキー無表示
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z    ←すべての変数の整数宣言
```

CLS文
WIDTH文

なお、普通プログラムでは先頭に画面クリアのための**CLS(シーエルエス)文**を入れるのですが、**WIDTH(ウィズ)文**を実行すると一度画面はクリアされるので省きました。

さて、シーケンシャルファイルに数値とか文字列を書き込むときには、**WRITE（ライト）文**を使います。書式は次のとおりです。

WRITE(ライト)文

WRITE #<ファイル番号>，<式>

WRITEとは、英語で「書く」という意味です。つまり**「ファイル番号で示されたファイルに、〈式〉で示された数値とか文字列を書き込む」**ということです。

書き込む

「書き込む」というと、鉛筆で紙に文字を書いたりすることを思い出しますが、ディスクの場合もこれと同じですが、「書く」というよりは**「記録する」**くらいの意味に考えておいたらよいでしょう。

〈注〉シーケンシャルファイルへの書き込みにはPRINT#（プリントシャープ）文も使えますが、WRITE文のほうがメモリ効率がよいのでこちらを使います。

いま述べたように**「式」には数値式あるいは文字式が入ります。**

また**「ファイル番号」には1から15までの整数を指定できますが、**システムディスクを起動させたときに画面に表示された

How many files(0−15)?

というメッセージがありましたが、**このとき入力した数値をこえてはいけません。**「同時にいくつファイルを使いますか？」というあなたへの問いかけでしたね。

〈注〉ファイル番号の指定はメッセージによれば（0－15）となっていますから0から15までの指定が許されるわけですが、0番は特殊な場合に使われますので、実際には1から15までのいずれかを指定することになります。

それではさっそく書き込み部分のプログラムを作ってみましょう。次のようになります。

```
170 INPUT "エイゴ ヲ ドウゾ"   ;A$   ─英単語と日本語の入力
180 INPUT "      ニホンゴ ハ ";B$
190 WRITE #1,A$,B$ ←────ディスクへデータを書き込む
```

さあ、［f・5］キーを押してRUNさせてみてください。適当な英単語と日本語訳を入力してみましょう。

おや？ ピーッというブザー音とともにエラーメッセージが返ってきました。

次のような画面になってコマンド待ちになってしまいました。

```
エイゴ ヲ ドウゾ  ? airplane   ─どちらも入力後リターンキーを押す
     ニホンゴ ハ  ? ヒコーキ
File not open in 190 ←────エラーメッセージ!!
Ok
■
```

File not open in 190

というエラー、つまりこれは直訳すると「**190行でファイルがOPENされていない**」ということです。ファイルがOPENされていないとはどういうことでしょうか。

実はファイルというのは、たとえてみればノートのようなもので、**開いてなければ書き込みができない**のです。

あなたがノートに何か書くとき、開いてから字を書きますね。ディスクの読み書きも原理はこれとまったく同じなのです。

ここで再び、シーケンシャルファイルとランダムファイルの違いについてふれておきます。

シーケンシャルファイルとランダムファイルの違い

ディスクをノートにたとえますと、シーケンシャルファイルの場合、1ページ目から順序よく書いていかなければならないのです。一方、ランダムファイルの場合は、自分の好きなページに、つまり、10ページ目でも50ページ目でも自由に書き込むことができるのです。

また、すでに書き込んであるノートのあるページの文章だけを読みたくなったとき、シーケンシャルファイルですと、自分の意志とは関係なく、1ページ目から、1、2、3、……と順序よくページをめくっていかなければならないのです。

ところが、ランダムファイルの場合、ノートのインデックスを思い出していただけばわかりやすいと思いますが、直接自分の希望するページを開いて読むことができるのです。ずいぶん便利ですね。

これで2つのファイルの違いがイメージ的に以前よりはっきりしてきたのではないかと思います。

ファイルのOPENとCLOSE

さて、画面にエラーメッセージが出たままストップしてしまいましたが、いまのたとえ話でもおわかりいただけたように、**ファイルという名のノートをまずOPEN(オープン)する、つまり開かなくてはいけないのです**。そして、これも大切なことですが、**書き終えたら閉じておく、つまりCLOSE(クローズ)しなければならないのです**。OPEN文は常にCLOSE文と対で使われます。

OPENする
OPEN文
CLOSEする
CLOSE文

書式は次のとおりです。

```
OPEN “ファイル名” FOR (モード) AS #(ファイル番号)

CLOSE #(ファイル番号), #(ファイル番号), ・・・・・・・
```

ファイル名は、そのファイルがドライブ1にあるのでしたらドライブ番号は省略できますが、ドライブ2にあるファイルに対しては次のようにドライブ番号をつけるのでした。

```
2: eitan1. dat
```

OPEN文の**ファイル番号**は0から15までの値を指定することができますが、39ページで説明したように、起動時に指定したファイル数をこえてはいけません。 **ファイル番号**

次に**モード**ですが、シーケンシャルファイルの場合、**アクセスの方法によって3種類あります。** **モード**

OUTPUT

```
OPEN "2: eitan1. dat" FOR OUTPUT AS #1
```

これはOUTPUT(**アウトプット**)**モード**と呼ばれ、数値あるいは文字データをディスクに書き込むときに使います。 **アウトプットモード**

2つ目はINPUT(**インプット**)**モード**です。 **INPUT**

```
OPEN "2: eitan1. dat" FOR INPUT AS #1
```

インプットモードはディスクに書き込まれているデータを読み出すときに使います。 **インプットモード**

そして最後がAPPEND(アペンド)モードです。 **APPEND**

```
OPEN "2: eitan1. dat" FOR APPEND AS #1
```

アペンドモードは、すでにディスク上に作成されているシーケンシャルデータファイルの末尾にデータを追加して書き込みたいときに使用します。 **アペンドモード**

以上3つのモードがありますが、これから実際に使いながら体得していくことにしましょう。

さて現在私たちが必要としているのはデータ書き込みですから**アウトプットモード**ということになります。ステートメントは次のようになります。

```
165 OPEN "2:eitan1.dat" FOR OUTPUT AS #1   ← ファイルを書き込みモードでオープン
200 CLOSE #1   ← 使用したファイルを閉じる
210 END
```

さあ、これで実行できるでしょうか。**RUNコマンド**で走らせてみてください。

```
ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ   ? airplane
     ﾆﾎﾝｺﾞ ﾊ   ? ﾋｺｰｷ
Ok
█
```

確かに成功したようですが、１単語登録しただけで終ってしまいました。もう少しデータ数をふやすことはできないでしょうか。

プログラムに、いくつかステートメントを追加してみましょう。次に追加後のプログラム総リストを示しますので、これを見ながら説明を読んでください。**緑色の行が追加ステートメント**です。

```
100 ' file name ･･･> eitan1.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ｶｷｺﾐ
130 ' save "eitan1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
165 OPEN "2:eitan1.dat" FOR OUTPUT AS #1
167 LOCATE 0,3
168 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"    ←このメッセージは常に表示
169 LOCATE 0,0
170 INPUT "ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ   ";A$
175 IF A$="999" THEN GOTO 200     ←入力終了？
180 INPUT "     ﾆﾎﾝｺﾞ ﾊ   ";B$
190 WRITE #1,A$,B$
195 LOCATE 0,0 : PRINT SPC(50) ┐
196 LOCATE 0,1 : PRINT SPC(50) ┘←英語、日本語の入力行を消す
197 GOTO 169
200 CLOSE #1
210 END
```

167、168行は、オペレータに終りのときはどうしたらよいかを知らせるメッセージの表示です。この場合「**999**」を入力してリターンキーを押すと終りとなるようにしました。

LOCATE文

169行の**LOCATE（ロケート）文**は、これを入れておかないと、168行のメッセージの下に「英語をどうぞ」という**プロンプト文**が表示されてしまいます。

175行は終りのマークである「999」が入力されたかどうかのチェックですね。終りの場合には、ファイルをCLOSE(クローズ)しなければなりませんので200行へ飛ばします。

195、196行は、

エイゴ　ヲ　ドウゾ　？ airplane
ニホンゴ　ハ　？ ヒコーキ

という**プロンプト文**とキーインされた英語、日本語を、1回ごとにクリアするためにスペースを出力させています。

プロンプト文

そして197行で169行へ飛ばしてループします。つまりこれで、英単語の入力に対して「999」がキーインされなければ何回でも繰り返せるというわけです。

古いファイルはキャンセルされる

さて、それでは実際にどんなふうになるか実行してみますが、その前に注意があります。私たちは先ほど修正前のプログラムで、

airplane
ヒコーキ

の2つの語を入力し、ドライブ2側のディスク上に

eitan1. dat

という**データファイル**を作りました。そこで今度、修正後のプログラムを実行すると、やはり前と同じように

データファイル

eitan1. dat

をOPEN（オープン）して（開いて）書き込むことになりますから、すでに書き込まれている単語の次に書き込まれると考えたくなりますが、**OUTPUT モードの場合**は、オープンしたファイルの内容はキャンセルされて新しくデータが書き込まれます。つまり、最初からやりなおしということになります。

ですから、前に、いくらたくさんのデータを書き込んであっても、このOUTPUTモードでファイルを開く（OPENする）と前のデータは無視され、新しく入力したデータが優先されます。

それでは追加するときはどうしたらよいのか、ということが問題になりますが、これはあとで解決しますのでここまでにして先へ進みましょう。

プログラムをRUN

それではプログラムを実行してみましょう。 f・5 キーを押してRUNさせてください。

次の２つの単語を登録してみましょう。

airplane　**ヒコーキ**
apple　**リンゴ**

次に示すのは入力中の画面です。

```
エイゴ゛ ヲ ト゛ウゾ゛  ? airplane
     ニホンゴ゛ ハ  ? ヒコーキ■

オワリ ノ トキ ハ 999 ヲ ト゛ウゾ゛
```

２つの単語を入力して、３回目に英語をどうぞになったとき「999」を入力してリターンキーを押すと次の画面になって処理終了です。

```
エイゴ゛ ヲ ト゛ウゾ゛  ? 999
Ok
■
オワリ ノ トキ ハ 999 ヲ ト゛ウゾ゛
```

この画面表示になれば、一応データの書き込みはできたと考えてよいと思いますが、念のためドライブ２側のディスクの内容を表示させてみましょう。**FILES（ファイルズ）コマンド**を使えばよかったですね。

FILESコマンド

```
files 2
eitan1 dat  1
Ok
■
```

LFILESコマンド

●ファイル名をプリンタに打ち出したいときはFILESの前にLをつけて
　LFILES
としてください。読み方は「エルファイルズ」です。

どうですか？
確かにデータファイル

eitan1. dat

が作られていますね。まあ、とりあえず安心というところです。

登録数の表示

しかし、それにしてもフィニッシュが少々味気ないですね。もう少しなんとかしたいものです。

登録した単語をかぞえて（**カウント**して）、その数を画面に表示して終るようにしましょう。　カウントする

カウンタを入れる箇所ですが、書き込みが終った時点で入れるとよいと思います（40ページのリスト参照）。次を入力しましょう。　カウンタ

```
192 C=C+1
```

このステートメントによって、英単語、日本語をペアとして登録するごとに登録数が1ずつ加算され、それが変数Cに代入されます。そして、これをEND文の前で画面に表示すればよいわけです。

次のプログラムを入力してください。

```
205 CLS
206 PRINT "トウロク ソウスウ";C;"タンコ゛ テ゛ス"
```

さあ、これで必要なステートメントは追加できました。いくつかサンプルを入力してみましょう。

プログラムをRUNさせて、次の10単語を登録してみてください。

日本語訳を入力するときは カナ キーをロックしなければなりません。英語、日本語の入力に対して、いちいち カナ キーの切り換えを行わなければならないので少々めんどうですが、慣れてしまえばどうということはありません。

❶ airplane・・・・ヒコーキ	❷ apple ・・・・リンゴ
❸ bee・・・・・・・ミツバチ	❹ bicycle ・・・ジテンシャ
❺ blackboard・・コクバン	❻ castle・・・・オシロ
❼ city・・・・・・・トシ	❽ college ・・・ダイガク
❾ comic book・・マンガ	❿ desk・・・・・ツクエ

どうですか？　うまくいきましたか？　10単語の入力を終え、11個目の単語入力に対して「999」を入力すると、画面に

```
トウロク ソウスウ 10 タンコ゛ テ゛ス
```

と表示されて終わればOKです。

RENUMコマンド

プログラムの行番号がふぞろいですので、**RENUM(リナンバ)コマンド**で整理しておきましょう。

```
renum 100 ⏎
```

そして**プログラムのセーブ**です。

```
save "eitan1.bas" ⏎
```

最後に、プログラムの総リストを示しておきます。

```
100 ' file name ····> eitan1.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ｶｷｺﾐ
130 ' save "eitan1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:eitan1.dat" FOR OUTPUT AS #1
180 LOCATE 0,3
190 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
200 LOCATE 0,0
210 INPUT "ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ   ";A$
220 IF A$="999" THEN GOTO 290
230 INPUT "     ﾆﾎﾝｺﾞ ﾊ   ";B$
240 WRITE #1,A$,B$
250 C=C+1
260 LOCATE 0,0 : PRINT SPC(50)
270 LOCATE 0,1 : PRINT SPC(50)
280 GOTO 200
290 CLOSE #1
300 CLS
310 PRINT "ﾄｳﾛｸ ｿｳｽｳ";C;"ﾀﾝｺﾞ ﾃﾞｽ"
320 END
```

ディスクに書くとはどういうこと?

私たちはこれで、

eitan1.dat

というデータファイルに10組の英単語・日本語を書き込みました。そして、鉛筆でノートに何かを書くかのように「ディスクに書き込む」という表現を使ってきました。

では「書く」とはいったいどういうことでしょうか?

答から先に申しますと、私たちがキーボードから入力したデータはコンピュータ内部で電気信号に変換され、それがディスクという磁気盤に記録されているのです。

記録されているといっても、目に見えないからタチが悪いですね。私たちの希望どおりにうまく記録できたのかどうか、まったく自信がもてません。

これは、ディスクを勉強し始めたころ誰でもがいだく疑問です。しかし、こういった疑問も、先へ進むにつれて徐々に解決していくと思いますので、ご安心ください。

大切なことは、最初疑問が生じるかもしれませんが、あまり深く考えつめないで、正しい答さえ得られたならば、それはそのままにして先へ進むことです。

2·5　ディスクのデータを読む

この節では、前節でディスクに書いたデータを読んで、それをディスプレイに表示してみます。実際に書き込みに成功したかどうかまだ不安です。はやく確かめたいものですね。

インプットモード

データを読むときも先ほどのノートの考え方と同じです。つまり、**ノートを開いて（OPENして）、読み終ったら閉じる（CLOSEする）**のです。ただしこの場合、モードが読み出しのモード（**インプットモード**）になります。形は次のとおりです。

```
OPEN "2:eitan1.dat" FOR INPUT AS #1
```

（INPUT：インプットモード）

このステートメントの最後に1というファイル番号がついていますが、これについて補足説明しておきます。

バッファ

パソコンでディスクへの読み書きをする場合、実は直接行うのではなく**本体内のバッファという場所を中継して行う**のです。このバッファエリアの数はディスク起動時の

```
How many files(0-15)?
```

で指定した数だけ確保されます。

上のメッセージでは(0－15)となっていますから**16のバッファエリアを使うことが可能**ですが、実際のプログラムでファイルを同時にこんなにたくさんOPENすることもありませんし、また**1つのバッファで256バイト必要**ですから、メモリもそれだけよけいにとられてしまいます。**〈注〉**

〈注〉メモリをよけいとられるといっても最近はずいぶん容量もふえていますのでたいしたことはありませんが……。

普通は5くらいを入力しておいて、この範囲内でうまくやりくりするほうがよいでしょう。

たとえば、1つのプログラムで6つのファイル（仮にA、B、C、D、E、Fとしておきます）を使うとしても、＃1のバッファエリアをAに、＃2のバッファエリアをBに、＃3のバッファエリアをCに、というようにいつも限定して割り当ててしまうのではなく、たとえば、＃2のバッファエリアを使っているファイルBをCLOSEすれば、ここを残りのA、C、D、E、Fは自由にOPENして使うことができるのです。つまり、5つのバッファエリアを同時に6つ以上のファイルの入出力で使いさえしなければ、どう使おうと自由なのです。

プログラムの流れの中で、使ってないファイルはCLOSEして、バッファエリアをほかのファイルに貸してあげればよいでしょう。

また、どんなにデータ量がぼう大であっても、1つのバッファには**256バイト**の広さしかありませんから、これを単位として読み書きを行っています。

バッファについては少々わかりにくいかもしれません。これから徐々に明らかになりますから、いまは、**データの書き込み・読み出しには中継点が必要**だと考えておいてください。

話しをもとに戻しましょう。

インプットモードでファイルをOPENしましたら、次はデータを読まなければなりませんが、これには**INPUT#(インプットシャープ)文**を使い、その書式は次のようになります。

INPUT#文

〈注〉読み方は「インプットシャープ」でも「インプットナンバー」でもどちらでもかまいません。

INPUT　#(ファイル番号),(変数),(変数),　………

ファイル番号は1から15まで指定できますが、ディスク立ち上げ時のHow many files? に対してキーインした数字を超えてはいけません。

また、**変数**は数値変数でも文字型変数でも、どちらでもかまいません。並数並びはコンマで区切ってください。

ファイルの読み出しは書き込みの逆ですからプログラムの組み立ては同じです。つまり、**ファイルをINPUTモードでOPENし、次にINPUT#文により読み出す**わけです。

〈注〉プログラム処理終了のしるしであるEND文があればOPENしているファイルはすべて自動的にクローズされるので、下のリストの210行のCLOSE文は不要ですが、形を覚えていただくために入れておきました。

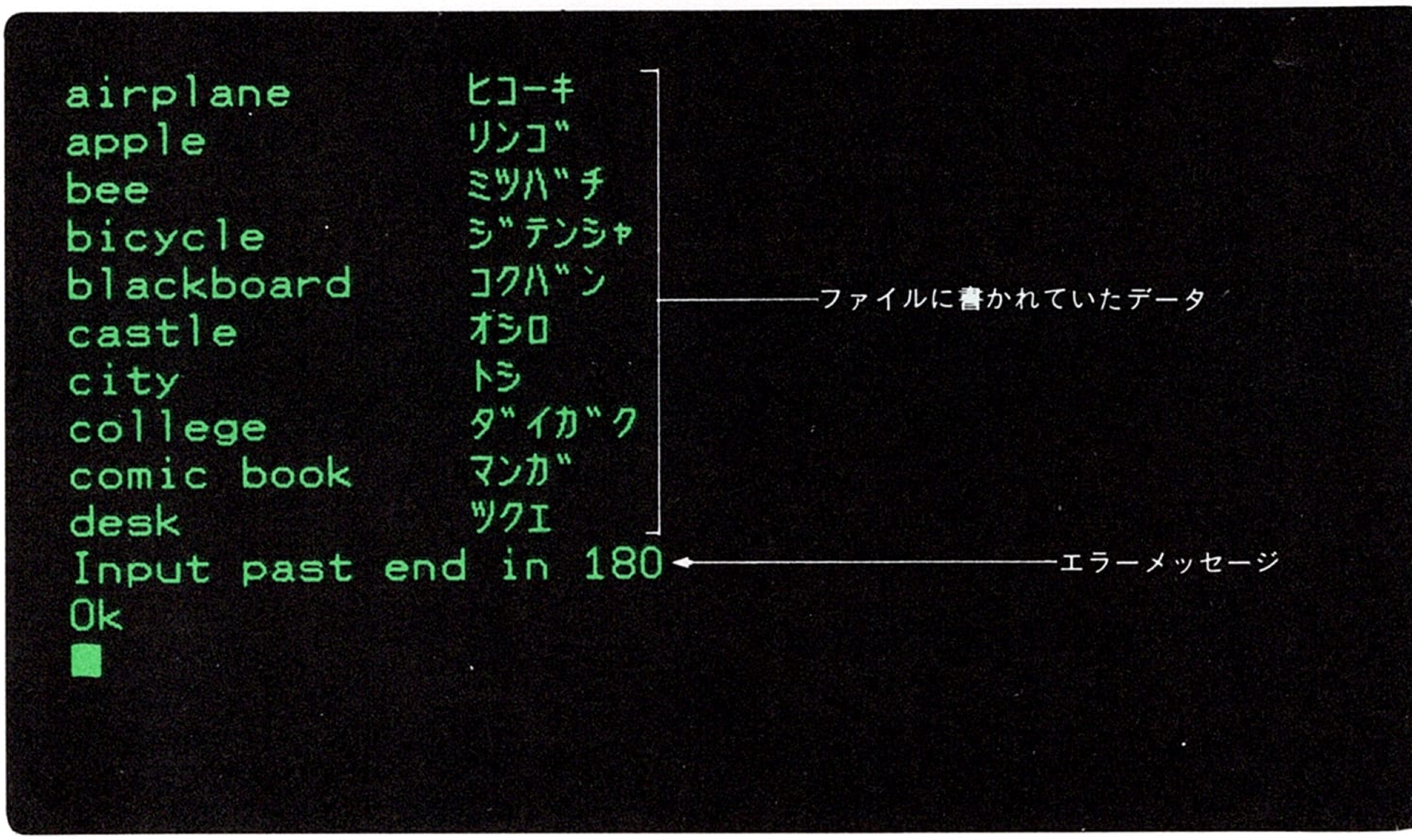
プログラム名を**eitan2.bas**としてデータ読み出しのプログラムを作ってみましょう。リストは次のとおりです。

```
100 ′ file name ･･･> eitan2.bas
110 ′ 1985.2.15
120 ′ ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ﾖﾐﾀﾞｼ
130 ′ save "eitan2.bas"
140 ′
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:eitan1.dat" FOR INPUT AS #1
180 INPUT #1,A$,B$
190 PRINT A$,B$
200 GOTO 180
210 CLOSE #1
220 END
```

- eitan2.bas ← プログラム名
- 130行 ← プログラムセーブ用REM文
- 170行 ← ファイルオープン
- 180行 ← データを読む
- 190行 ← ディスプレイに表示
- 210行 ← ファイルを閉じる

前節で書き込んだデータを読んでディスプレイに表示するようにしてあります。45ページのデータは書き込んでありますか？

それではRUNしてみてください。
おやおや、またピーッというブザー音とともにOkの表示がでてコマンド待ちとなりました。



このエラーメッセージ、

Input past end in 180

は、ディスクを扱うようになると、初めの頃よく出会うメッセージです。意味は**「行番号180で(ファイルの)終りを過ぎているのにまだ(データを) 読もうとしている (INPUT しようとしている)」**といったところです。

プログラムを見てください。

180行でデータを読んで190行で画面に表示し、200行で再び読み込み処理へ飛ばしています。これではファイルにデータがなくなってもまだ読み込もうとしてしまうはずです。どこかでファイルの末尾をチェックさせて、このループから抜け出さなければなりません。

EOF関数

●読み方は「イーオーエフ」または「エンド・オブ・ファイル」と読みます。

この、**データチェックのための関数がEOF**(End of fileの略)です。コンピュータは、**この関数の値が−1 (真、つまりファイルの末尾であるということ) か0 (偽、つまりまだファイルがあるということ) かによって判断**しているのです。

ですから、この関数をデータを読むステートメントの前に入れなけ

ればなりません。

```
175 IF EOF(1) THEN GOTO 210
```

EOF(1)、つまり＃1のファイルがもしEnd of fileならば、ファイルのCLOSEをします。そして、行番号200のGOTO文を変更して、125行へ飛ばすようにします。

〈注〉EOF関数のカッコの中の数字は読み出しのためにOPENしているファイル番号が入ります。

```
200 GOTO 175
```

プログラムを実行してみましょう。今度はエラーもなく無事に単語を画面に表示できたと思います。

問答形式にする

しかしこれでは、ディスクからデータを読んで、それをディスプレイに表示したというだけで、英単語を表示し、それに対して答えるという形式がととのってぃません。少し手直ししましょう。

手直ししたプログラムを次ページに示します。緑色の部分が追加行および修正箇所です。

172、173行は画面の4行目（一番上は0行になっています）に**オペレーション**のためのメッセージを表示します。

オペレーション

181〜183行は、英単語を画面に表示したことに対して注目させるための**ビープ（ブザー）音**です。ループ変数の最終値200を変更することによって、長さを調節できます。

ビープ音

190行で英単語を出力し、192行でその意味を問いかけます。ここで注意することは、192行の最後部に**セミコロン**をつけることです。これをつけ忘れると、194行で日本語を出力するときに続けて表示されず改行されてしまいます。

セミコロン

193行は**INKEY$（インキードル）関数**を使ったキー入力待ちのステートメントです。この本の前編で勉強したからわかりますね。

時間待ち（タイマー）

INKEY$関数

●インキーダラと続んでもかまいません。

195行は**時間待ち（タイマー）**のための常とう手段です。これがないと、次に**SPC（スペース）関数**があるので瞬間的に消去され、日本語が表示されたのかどうかわかりません。

SPC関数

196、197行ではLOCATE（ロケート）文のX座標を15とし、英単語、日本語訳だけを消去します。

ちょっと退屈な勉強をしたのですから、なぐさめの言葉を最後に表示させましょう。これは216行で行っています。

```
100 ' file name ····> eitan2.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ﾖﾐﾀﾞｼ
130 ' save "eitan2.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:eitan1.dat" FOR INPUT AS #1
172 LOCATE 0,3
173 PRINT "ｺﾀｴﾀﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
175 IF EOF(1) THEN GOTO 210
180 INPUT #1,A$,B$
181 FOR I=1 TO 100
182   BEEP 1 : BEEP 0
183 NEXT I
185 LOCATE 0,0
190 PRINT "ｴｲｺﾞ··········> ";A$
192 PRINT "ｲﾐ ﾊ ? ······> ";
193 I$=INKEY$ : IF I$="" THEN GOTO 193
194 PRINT B$
195 FOR I=1 TO 1000 : NEXT I
196 LOCATE 15,0 : PRINT SPC(50)
197 LOCATE 15,1 : PRINT SPC(50)
200 GOTO 175
210 CLOSE #1
215 CLS
216 PRINT "ｵﾂｶﾚ ｻﾏ"
220 END
```

上のプログラムリストの追加行を入力し、修正部分の手直しはすみましたか？　できたら実行です。

f・5 キーを押してRUNしてみてください。次のような画面になり、あなたが「airplane」に対する日本語の意味を答えるのを待っています。

```
ｴｲｺﾞ··········> airplane
ｲﾐ ﾊ ? ······>

ｺﾀｴﾀﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ
```

答えたらリターンキーを押してください。次の単語に変わります。
10語答えると次の画面になって終りです。

```
オツカレ サマ
Ok
■
```

さあ、それではプログラムの行番号がふぞろいですので行番号を整理しましょう。
RENUMコマンドを使えばよいですね。

```
renum 100 ⏎
```

そして、忘れずにこのプログラムをセーブしておきましょう。

```
save "eitan2.bas" ⏎
```

最後に、本節でのプログラムリストを示しておきます。

```
100 ' file name ····> eitan2.bas
110 ' 1985.2.15
120 ' シーケンシャル データ ノ ヨミダシ
130 ' save "eitan2.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:eitan1.dat" FOR INPUT AS #1
180 LOCATE 0,3
190 PRINT "コタエタラ RET キー ヲ ドウゾ"
200 IF EOF(1) THEN GOTO 340
210 INPUT #1,A$,B$
220 FOR I=1 TO 100
230   BEEP 1 : BEEP 0
240 NEXT I
250 LOCATE 0,0
260 PRINT "エイゴ··········> ";A$
270 PRINT "イミ ハ ? ······> ";
```

```
280 I$=INKEY$ : IF I$="" THEN GOTO 280
290 PRINT B$
300 FOR I=1 TO 1000 : NEXT I
310 LOCATE 15,0 : PRINT SPC(50)
320 LOCATE 15,1 : PRINT SPC(50)
330 GOTO 200
340 CLOSE #1
350 CLS
360 PRINT "ｵﾂｶﾚ ｻﾏ"
370 END
```

2・6 データを追加する

これまで私たちは、ディスクにデータを書き込み、それを読み出して画面に表示するところまで学習してきました。

データ量が多くて、とてもひと息に入力できないとか、何かの都合で次回にまわしたいとか、いろいろ事情も生じてくると思います。

残りのデータを現在作成されているデータファイルに後日あらためて追加して書き込みたいという場合はどうしたらよいでしょうか。

OUTPUTモードではデータ書き込みはできましたが、このモードでデータファイルをOPENすると、すでに書かれているデータがご破産になり、あらためて最初から入力することになります。つまり、苦心して書き込んだデータが消えてなくなってしまうのです。

ファイルポインタ

これはメモリの中に**ファイルポインタ**というものがあって、ファイルをOUTPUTモードでOPENすると常にこのポインタがファイルの先頭を指し示し(セットされる)、新規に書き込みを行うことになるからです。初めのうちは、このことを知らずにデータを消失してしまうこともありますので、注意してください。このようなミスを防ぐ方法は、のちほど説明します。

INPUTモードでもファイルポインタはファイルの先頭にセットされますが、読み出しモードですので心配ありません。

データをシーケンシャルファイルに追加して書き込みたい場合は、**APPEND(アペンド)モード**を使います。

```
OPEN "2:eitan1.dat" FOR APPEND AS #1
```

(APPEND ← データ追加のモード)

すでにディスクの中にある「eitan1.dat」というデータファイルを1番のバッファを使ってAPPENDモードでOPENしなさい、という意味です。

そして、このOPEN文が実行されると、ファイルポインタはファイルの末尾にセットされ、追加して書き込みできる準備が完了したことになります。

APPENDモードはOUTPUTモードと同じように、データを書き込むという点においては変わりありませんから、その他のステートメントは同じで、OUTPUT用のプログラムをスクリーンエディタで多少変更するくらいで使うことができます。

それでは書き込み用のプログラムファイル「**eitan1.bas**」をロードして、**スクリーンエディタ**で修正してみましょう。

```
load "eitan1.bas" ⏎
```

LISTコマンド

プログラムリストを次に示します。みなさんは**LIST（リスト）コマンド**で画面に表示してみてください。

```
100 ' file name ····> eitan1.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ｶｷｺﾐ
130 ' save "eitan1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:eitan1.dat" FOR OUTPUT AS #1
180 LOCATE 0,3
190 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
200 LOCATE 0,0
210 INPUT "ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ   ";A$
220 IF A$="999" THEN GOTO 290
230 INPUT "      ﾆﾎﾝｺﾞ ﾊ   ";B$
240 WRITE #1,A$,B$
250 C=C+1
260 LOCATE 0,0 : PRINT SPC(50)
270 LOCATE 0,1 : PRINT SPC(50)
280 GOTO 200
290 CLOSE #1
300 CLS
310 PRINT "ﾄｳﾛｸ ｿｳｽｳ";C;"ﾀﾝｺﾞ ﾃﾞｽ"
320 END
```

さて、上のリストはデータ書き込みのときのプログラムです。これを一部修正してデータ追加のプログラムにしようというわけです。それには5箇所**スミアミ**になっているところがあります。ここを変更すればよいのです。

プログラム名は

eitan3. bas

とします。

カーソルを移動させて、5つの行の一部を次のように変更してください。緑色で示してあります。

```
100 ' file name ····> eitan3.bas
120 ' シーケンシャル データ ノ ツイカ
130 ' save "eitan3.bas"
170 OPEN "2:eitan1.dat" FOR APPEND AS #1
310 PRINT "ツイカ ソウスウ";C;"タンゴ デス"
```

スクリーンエディタ

たとえば行番号100の修正でしたら、

LIST 100 ⏎

として100行のステートメントを画面に表示させ（すでに表示されていればそれには及びません）、カーソルを「1」の上に重ね 3 キーを押して ⏎ キーを押します。これで行番号100の修正は終りです。

同様の方法で修正を行ってください。

ここでちょっと横道にそれます。

私たちはよく**スクリーンエディタ**という言葉を口にしますが、これは一種のソフトウェアなのです。

私たちは何げなくカーソルを移動して、追加・修正・削除などを行い、リターンキーを押して新しいステートメントにしていますが、こんなに簡単にできるのも、このスクリーンエディタのおかげなのです。これに対して**ラインエディタ**というのがあり、これは、ある箇所を修正しようとしても、スクリーンエディタのようにはうまくいかず、もう一度その行（ライン）を入力しなおすといった不便さです。

さて修正が終りましたら、とりあえず**このプログラムをセーブ**しておきましょう。行番号130のステートメントを使うと便利です。

まず、

LIST 130 ⏎

として、130行を画面に表示します。LISTは f・4 キーに定義されていますからこれを使ってください。

```
list 130
130 '  save "eitan3.bas"
Ok
■
```

上のような画面になりましたら、カーソル移動キーを使ってカーソルを「**save**」の「**s**」の上に重ねます。

```
list 130
130 '  save "eitan3.bas"
Ok
```

カーソルをここに移動し DEL キーを6回たたく

次に DEL キーを6回たたくと「s」の前の6字分が削除されて「s」が行頭にきますから、ここで ⏎ キーを押してください。

すると、ドライブ1側のアクセス表示用LEDが赤く点灯して、たちどころにプログラムがセーブされます。

データを追加してみよう

セーブできました。これで安心です。

それではさっそくためしてみましょう。次の6つの基本単語を追加することにします。カナ キーの切り換えを正確にしてください。

❶ **friend・・・・・・トモダチ** ❷ **iris・・・・・・アヤメ**
❸ **kindergarten・・ヨウチエン** ❹ **lily・・・・・・ユリ**
❺ **map・・・・・・・チズ** ❻ **pear・・・・・ナシ**

f・5 キーでRUNさせてください。

登録の方法は書き込みのときと同じですからおわかりと思います。6つの単語を登録し「999」をキーインすると次の画面になって追加処理の終了です。

```
ﾂｲｶ ｿｳｽｳ 6 ﾀﾝｺﾞ ﾃﾞｽ
Ok
■
```

画面に Ok がでています。しかし、これはシーケンシャルファイルへのデータの追加が確実にできたことの保証ではありません。パソコンは処理が終わればいつも Ok サインをだすのですから……。まだまだ安心できません。前節で作った読み出しのプログラムで確かめてみましょう。

ここで1つ新しい知識をマスターしておきましょう。いままではプログラムを実行するのに、まずロードして f・5 キーを押していましたね。これを一挙にやってしまう楽な方法があります。2つ示しますから、どちらの方法でもかまいません、キーイン後 ⏎ を押してみてください。

Rオプション付きのロード

```
❶ load "eitan2.bas",r ⏎
```

← Rオプション付つきのロード

```
❷ run "eitan2.bas" ⏎
```

❶も❷もまったく同じ機能です。これが実行されると、次々と英単語が表示されますから、声を出して答えたら ⏎ キーを押してください。最終画面は次のようになります。

```
オツカレ　サマ
Ok
■
```

仕事内容を表示する

データ追加のプログラムを実行するとおわかりのように、画面に突然データ入力をうながすメッセージがでてきて、ちょっととまどいを感じる方もいらっしゃるかもしれません。もう少し余裕をもたせてみましょう。要するに、もう少しパソコンと取り組んでいるという感じを持たせたいものです。

1つは、しばらく画面に**「データ　ヲ　ツイカ　シマス」**というメッセージを入れること。

もう1つは、**最終登録単語と登録語数を表示し、それを確認してから追加処理に入る**ことです。

それでは始めますが、現在メモリ上にはデータ読み出しのプログラムがロードされていますから、先ほどセーブしたデータ追加のプログ

ラムをもう一度呼び出してください。

```
load "eitan3.bas"
Ok
■
```

単語追加の作業に入る前に画面に次のような表示をすることにしましょう。

```
デ゛ータ ノ ツイカ ヲ シマス

マツビ゛ ノ タンコ゛・・・> pear
トウロク ソウスウ・・・・・> 16 コ゛

OK ナラ RET キー ヲ ト゛ウソ゛
```

さて、上の画面表示を作るのに3段階に分けます。

❶「データ　ノ　ツイカ　ヲ　シマス」を表示する。
❷ データファイルeitan1.datをインプットモードでOPENして、末尾の単語および登録総数を知る。
❸ 末尾の単語および登録総数、それに「OKナラRETキーヲ　ドウゾ」を表示する。

サブルーチンの使い方

これらはいずれも**サブルーチン**で処理することにします。サブルーチン形式にしなくともプログラムはできますが、ステートメントを追加するために現在の10きざみの行番号を変えたりしなければならないのでめんどうです。サブルーチンには、何回も使う部分をサブルーチン化する以外に、処理別に分類し、それをそれぞれサブルーチンにしてしまうという使い方もあります。

次のプログラムを入力してください。

```
165 GOSUB *SHIGOTO ←――― 仕事名表示
166 GOSUB *YOMU ←――― データの読み出し
167 GOSUB *HYOOJI ←――― それを表示する
```

まず、一番最初のサブルーチンです。プログラムリストを示しますから入力してください。

```
1000 ' ────────────────────────
1010 *SHIGOTO
1020 ' ────────────────────────
1030 PRINT "ﾃﾞｰﾀ ﾉ ﾂｲｶ ｦ ｼﾏｽ"
1040 FOR I=1 TO 3000 : NEXT I
1050 RETURN
1060 '
```

これはどこもむずかしいところはないでしょう。

行番号1000の長い横棒は、GRPH キーを押しながらテンキー側の * キーを押したとき表示される**グラフィックキャラクタ**です。40字分入っています。これは**REM（リマーク）文**ですから、みなさんの好きなようにしてください。マイナス記号とかスラッシュを使ってもよいでしょう。

1040行は時間待ちの**タイマー**です。　タイマー

さて次は❷の処理です。プログラムを次に示します。

```
2000 ' ────────────────────────
2010 *YOMU
2020 ' ────────────────────────
2030 OPEN "2:eitan1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   INPUT #1,A$,B$ ←───データ読み出し
2060   C=C+1 ←───カウンタ
2070 WEND
2080 CLOSE #1 : RETURN
2090 '
```

2050行のINPUT＃文はデータの読み出し、2060行はディスクに書き込まれているデータ数を数えるためにカウントしています。

2030行の＃１と2050行の＃１を同じファイル番号にしておいてください。あたりまえといえばあたりまえですが、プログラムにだんだん慣れてくると、この番号を間違えたがためにエラーになったり、あるいは、エラーにならなくとも正しい読み書きができなかったりということがよくあります。

さて、ここで新しいステートメントが登場しました。2040行、2070

行をごらんください。WHILEとWENDにはさまれていますね。この形の文を**WHILE～WEND（ホワイル～ウェンド）文**といいます。書式は次のとおりです。

WHILE～WEND
●ホワイル～ダブルエンドと読む人もいます。

```
WHILE  (条件)
  ∫
WEND
```

これは、**「条件」が成り立っている間はWHILE文とWEND文の間をループする**、大変便利なステートメントです。

NOT（ノット）
●否定を表す論理式で、たとえば「NOT　1」とすれば「1でない」という意味になります。

EOF
●ファイルの終りをチェックする関数。イーオーエフまたはエンド・オブ・ファイルと読みます。

2040～2070行の例でいきますと、次の条件、

NOT　EOF(1)

つまり、**「1番のファイルがEOFでない(NOT)」**ならば2050行のデータ読み出しと2060行の登録数加算の処理を行うことになります。

WEND文はWHILE文と対になり、ループの終りを示す働きをしますので省略できません。もし、つけ忘れたりすると、

WHILE　without　WEND

のエラーになります。

ファイルポインタ

2030行でファイルをインプットモードでオープンしますと、**ファイルポインタ**がファイルの先頭に行きます。そして、そのポインタで指し示されたところが**ファイルの終りでなければ**〈NOT　EOF(1)〉、英単語と日本語をセットにして2050行で読み出しますから、最終的にはA＄には末尾の登録英単語が代入されます。

RETURN文

登録データの読み飛ばしが終了すると、**RETURN文**でメインルーチンに戻るわけですが、インプットモードでOPENしたファイルを閉じる(**CLOSEする**)のを忘れないようにしてください。このモードでファイルをOPENした使命は果たしたわけですから、きちんと閉めておきましょう。これをおこたると、メインに戻って次に同じファイルをOPENしようとすると、

File　already　open

のエラーになります。「すでにファイルはオープンされていますよ」という警告です。

最後に３番目の処理を入力してしまいましょう。

```
3000 '────────────────────────────
3010 *HYOOJI
3020 '────────────────────────────
3030 PRINT
3040 PRINT "ﾏﾂﾋﾞ ﾉ ﾀﾝｺﾞ･･･> ";A$
3050 PRINT "ﾄｳﾛｸ ｿｳｽｳ･････>";C;"ｺ"
3060 PRINT
3070 PRINT "OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
3080 I$=INKEY$ : IF I$="" THEN GOTO 3080
3090 CLS
3100 RETURN
```

この部分は別に問題はないと思います。

RETURN文で戻る前に**CLS（クリアスクリーン）文**で画面をクリアしておきましょう。

CLS文

●CLSの読み方はシーエルエスでもかまいません。

次に、2060行で使ったカウンタ用変数Cはメインルーチンでも使いますので、内容をクリアしておく必要があります。

```
168 C=0
```

そして、次の１行を追加してプログラムを見やすくしましょう。

```
360 '
```

行番号がふぞろいの箇所がありますのでつけ変えておきましょう。

```
renum 100
Ok
renum 1000,410
Ok
renum 2000,1070
Ok
renum 3000,2130
Ok
█
```

そして最後に**このプログラムのセーブ**です。

```
save "eitan3.bas" ⏎
```

プログラムの実行

セーブできましたらRUNさせて、次の4単語を追加してみましょう。やり方は簡単ですから説明は省略します。

❶ **poster・・・・・ポスター**
❷ **seat・・・・・・・ザセキ**
❸ **student・・・・ガクセイ**
❹ **wolf・・・・・・オオカミ**

なお、プログラムの実行中に STOP キーを押してしまったとか、何かエラーメッセージで止まってしまったとかしたときは、ダイレクトモードで、次のいずれかを行ってください。

```
CLOSE ⏎
END ⏎
```

〈注〉END文を実行すると自動的に現在オープンしているファイルはクローズされます。

ファイルがOPENしたままである可能性もありますので、とりあえずCLOSEしておきます。こうしておけば、作成中のデータファイルを壊さずにすみます。

最後に総プログラムリストをかかげます。

```
100 ' file name ････> eitan3.bas
110 ' 1985.2.15
120 ' シーケンシャル データ ノ ツイカ
130 ' save "eitan3.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 GOSUB *SHIGOTO
180 GOSUB *YOMU
190 GOSUB *HYOOJI
200 C=0
210 OPEN "2:eitan1.dat" FOR APPEND AS #1
220 LOCATE 0,3
230 PRINT "オワリ ノ トキ ハ 999 ヲ ドウゾ"
240 LOCATE 0,0
250 INPUT "エイゴ ヲ ドウゾ   ";A$
260 IF A$="999" THEN GOTO 330
270 INPUT "      ニホンゴ ハ   ";B$
280 WRITE #1,A$,B$
290 C=C+1
```

```
300 LOCATE 0,0 : PRINT SPC(50)
310 LOCATE 0,1 : PRINT SPC(50)
320 GOTO 240
330 CLOSE #1
340 CLS
350 PRINT "ﾂｲｶ ｿｳｽｳ";C;"ﾀﾝｺﾞ ﾃﾞｽ"
360 END
370 '
1000 ' ------------------------------------------------
1010 *SHIGOTO
1020 ' ------------------------------------------------
1030 PRINT "ﾃﾞｰﾀ ﾉ ﾂｲｶ ｦ ｼﾏｽ"
1040 FOR I=1 TO 3000 : NEXT I
1050 RETURN
1060 '
2000 ' ------------------------------------------------
2010 *YOMU
2020 ' ------------------------------------------------
2030 OPEN "2:eitan1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   INPUT #1,A$,B$
2060   C=C+1
2070 WEND
2080 CLOSE #1 : RETURN
2090 '
3000 ' ------------------------------------------------
3010 *HYOOJI
3020 ' ------------------------------------------------
3030 PRINT
3040 PRINT "ﾏﾂﾋﾞ ﾉ ﾀﾝｺﾞ･･･> ";A$
3050 PRINT "ﾄｳﾛｸ ｿｳｽｳ････>";C;"ｺﾞ"
3060 PRINT
3070 PRINT "OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
3080 I$=INKEY$ : IF I$="" THEN GOTO 3080
3090 CLS
3100 RETURN
```

2·7 データ入力の確認

これで、シーケンシャルファイルへのデータ書き込み、追加、そして読み出すこともできるようになりました。

しかし追加の場合、英単語と日本語訳を入力するとすぐに登録されてしまい、間違ってキーインしても修正することができませんでした。

そこでこの節では、データの修正とはいきませんが、せめて確認ルーチンだけでも作っておくことにしましょう。前節でのデータ追加プログラムにステートメントを追加することにします。

まず、前節のデータ追加プログラムをロードしてください。

```
load "eitan3.bas" ⏎
```

ロードできたら、100行から320行くらいまでをLISTコマンドで表示させてみてください。

```
100 ' file name ····> eitan3.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ﾂｲｶ
130 ' save "eitan3.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 GOSUB *SHIGOTO
180 GOSUB *YOMU
190 GOSUB *HYOOJI
200 C=0
210 OPEN "2:eitan1.dat" FOR APPEND AS #1
220 LOCATE 0,3
230 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
240 LOCATE 0,0
250 INPUT "ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ   ";A$
260 IF A$="999" THEN GOTO 330
270 INPUT "      ﾆﾎﾝｺﾞ ﾊ  ";B$
280 WRITE #1,A$,B$
290 C=C+1
300 LOCATE 0,0 : PRINT SPC(50)
310 LOCATE 0,1 : PRINT SPC(50)
320 GOTO 240
```

前ページの画面を見てください。100行、130行のファイル名は、あとで、

eitan4. bas

と変更することにします。

さて、**データの確認ルーチンを入れる場所**ですが、英単語、日本語の書き込み前にこの処理をしなければなりませんから、270行と280行の間にステートメントを入れてサブルーチンに飛ばせばよいでしょう。まず、次の1行を入力してください。

```
272 GOSUB *KAKUNIN
```

データ確認のサブルーチンへジャンプさせます。

そしてサブルーチンの入力となりますが、わかりやすいように、どんな動作ステップをふむのか簡単に示しておきます。

まず、このプログラムをRUNさせたとすると、次の初期画面ができます。

```
デ゛ータ ノ ツイカ ヲ シマス

マツビ゛ ノ タンゴ゛・・・> wolf
トウロク ソウスウ・・・・・> 20 コ゛

OK ナラ RET キー ヲ ト゛ウゾ゛
```

リターンキーを押すと、次のような入力画面になります。

```
エイゴ゛ ヲ ト゛ウゾ゛  ? station
     ニホンゴ゛ ハ  ? エキ■

オワリ ノ トキ ハ 999 ヲ ト゛ウゾ゛
```

従来なら、上の画面のように英単語と日本語を入力しリターンキーを押すと、たとえ間違っていても登録されてしまい、シーケンシャルファイルに書き込まれてしまいましたが、ここで一度、次のように確

認させることにします。

```
エイゴ゛ ヲ ト゛ウゾ゛  ? station
    ニホンゴ゛ ハ  ? エキ

トウロク シマスカ ( y/n ) ?
```

ここで「Y」が入力されれば登録、「N」ならもう一度最初から正しいデータを入力することになります。

確認サブルーチン

それでは次に**サブルーチン**を示しますので入力してください。

```
4000 '―――――――――――――――――――――――
4010 *KAKUNIN
4020 '―――――――――――――――――――――――
4030 FOR I=1 TO 200          ┐
4040   BEEP 1: BEEP 0        ├――ブザー音
4050 NEXT I                  ┘
4060 LOCATE 0,3
4070 PRINT "トウロク シマスカ ( y/n ) ?    "
4080 I$=INKEY$ : IF I$="" THEN GOTO 4080
4090 IF I$="Y" OR I$="y" THEN GOTO 4120
4100 IF I$="N" OR I$="n" THEN GOTO 4120
4110 BEEP : BEEP : BEEP : GOTO 4080 ←――┐
4120 LOCATE 0,3              キー入力ミスのときの処理
4130 PRINT "オワリ ノ トキ ハ 999 ヲ ト゛ウゾ゛"
4140 IF I$="Y" OR I$="y" THEN RETURN
4150 RETURN 300 ←―――――――――「N」の場合
```

<注>4070行のメッセージに対してのキー入力は大文字でも小文字でもかまいません。また、これ以外のキー入力の場合にはビープ音が発生し、再度YまたはNの入力待ちとなります。

別にむずかしい部分はないと思います。

4130行は、4070行でメッセージ欄に

登録しますか（Y／N）？

を出力しましたので、メインルーチンへ戻る前にこれを変更するためです。

4140行は登録する場合です。このときは、272行でこのサブルーチンへジャンプしてきましたから戻り先は272行のすぐ後ろ、つまり行番号280が実行されますが、「N」のときは登録しないわけですから、メインルーチンへ戻っても280行の書き込みおよび290行のカウンタ

はパスさせなければなりません。こんなとき使われるのが**行番号指定のRETURN（リターン）文**です。4150行で行っています。　**RETURN文**

それでは、行番号100、130のファイル名をスクリーンエディタで変更し、このプログラムをセーブしておきましょう。

```
100 ' file name ････> eitan4.bas
130 ' save "eitan4.bas"
```

ファイル名は「3」を「4」に変更するだけです。カーソルを3に重ね、［4］キーを押して［⏎］です。

```
save "eitan4.bas" ⏎
```

セーブが終了しましたら、プログラムを実行して、次の5つの単語を追加入力してみましょう。

❶ station ････エキ
❷ beach･････ハマベ
❸ ship ･･････フネ
❹ tiger･･････トラ
❺ bird･･･････トリ

確認のための

トウロク　シマスカ（Y／N）　?

のメッセージに対して、Y、Nいずれの場合も正しく動作するかどうか、試してみてください。また、それ以外のキーも押してみてください。つまり、キー入力ミスの場合です。このときは**ビープ音**が発生するはずです。

次ページにプログラムの総リストをのせておきます。

```
100 ' file name ····> eitan4.bas
110 ' 1985.2.15
120 ' ｼｰｹﾝｼｬﾙ ﾃﾞｰﾀ ﾉ ﾂｲｶ
130 ' save "eitan4.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 GOSUB *SHIGOTO
180 GOSUB *YOMU
190 GOSUB *HYOOJI
200 C=0
210 OPEN "2:eitan1.dat" FOR APPEND AS #1
220 LOCATE 0,3
230 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
240 LOCATE 0,0
250 INPUT "ｴｲｺﾞ ｦ ﾄﾞｳｿﾞ    ";A$
260 IF A$="999" THEN GOTO 330
270 INPUT "      ﾆﾎﾝｺﾞ ﾊ   ";B$
272 GOSUB *KAKUNIN
280 WRITE #1,A$,B$
290 C=C+1
300 LOCATE 0,0 : PRINT SPC(50)
310 LOCATE 0,1 : PRINT SPC(50)
320 GOTO 240
330 CLOSE #1
340 CLS
350 PRINT "ﾂｲｶ ｿｳｽｳ";C;"ﾀﾝｺﾞ ﾃﾞｽ"
360 END
370 '
1000 ' ──────────────────────────
1010 *SHIGOTO
1020 ' ──────────────────────────
1030 PRINT "ﾃﾞｰﾀ ﾉ ﾂｲｶ ｦ ｼﾏｽ"
1040 FOR I=1 TO 3000 : NEXT I
1050 RETURN
1060 '
2000 ' ──────────────────────────
2010 *YOMU
2020 ' ──────────────────────────
2030 OPEN "2:eitan1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   INPUT #1,A$,B$
2060   C=C+1
```

```
2070 WEND
2080 CLOSE #1 : RETURN
2090 '
3000 ' ────────────────────────────────
3010 *HYOOJI
3020 ' ────────────────────────────────
3030 PRINT
3040 PRINT "ﾏﾂﾋﾞ ﾉ ﾀﾝｺﾞ･･･> ";A$
3050 PRINT "ﾄｳﾛｸ ｿｳｽｳ･････>";C;"ｺﾞ"
3060 PRINT
3070 PRINT "OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
3080 I$=INKEY$ : IF I$="" THEN GOTO 3080
3090 CLS
3100 RETURN
4000 ' ────────────────────────────────
4010 *KAKUNIN
4020 ' ────────────────────────────────
4030 FOR I=1 TO 200
4040   BEEP 1 : BEEP 0
4050 NEXT I
4060 LOCATE 0,3
4070 PRINT "ﾄｳﾛｸ ｼﾏｽｶ ( y/n ) ?   "
4080 I$=INKEY$ : IF I$="" THEN GOTO 4080
4090 IF I$="Y" OR I$="y" THEN GOTO 4120
4100 IF I$="N" OR I$="n" THEN GOTO 4120
4110 BEEP : BEEP: BEEP : GOTO 4080
4120 LOCATE 0,3
4130 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
4140 IF I$="Y" OR I$="y" THEN RETURN
4150 RETURN 300
```

2・8　メニュープログラムを作ろう

以上、「英単語練習プログラム」を例にとってシーケンシャルファイルについて説明してきました。

アウトプットモード、インプットモード、アペンドモード、いかがでしたか？　思ったほどむずかしくはなかったと思います。

これまでに私たちがドライブ1のディスクにセーブしたプログラムファイルは次の4つです。

eitan1. bas ← データ書き込み用（初回のみ）
eitan2. bas ← データ読み出し用
eitan3. bas
eitan4. bas ← データ追加用（eitan3. bas と eitan4. bas）

さて、今後あなたがこれらのプログラムを使おうとする場合、いまのままですと、どのプログラムがどんな目的で使われるかを覚えておいて、次のような操作によらなければなりません。

```
run "eitan1.bas" ⏎

run "eitan2.bas" ⏎

run "eitan3.bas" ⏎

run "eitan4.bas" ⏎
```

しかし、ファイル名を打ち間違えて、

```
File not found
```

というエラーメッセージが返ってきてギクッとしたり、いや、それよりとにかくキーインもめんどうです。もう少し楽をすることはできないでしょうか？

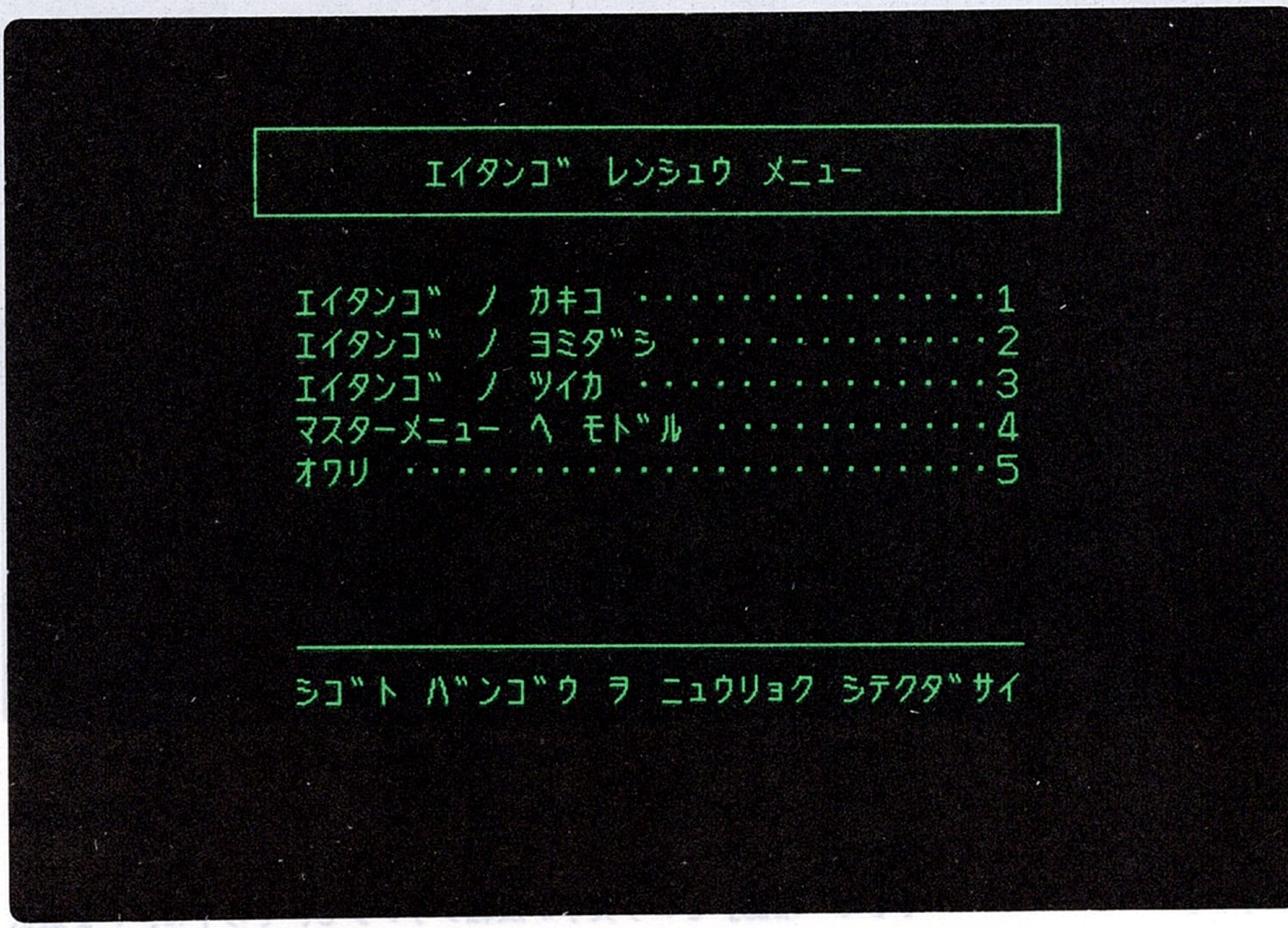
人間は1つのことをマスターすると、そのことをもう少し手をかけないでできないものかとムシのよい考え方をします。しかし、これはおおいにけっこうです。こういった欲がでてきてはじめて前進していくのですから……。

私たちもおおいに思いをめぐらせましょう。

そのためには、**「英単語練習プログラム」を管理するメニュープログラムを作ってしまう**ことです。 **メニュープログラム**

メニュープログラムとは、画面にいろいろな仕事内容を表示させ、その中からどれか1つを選び、希望する仕事を処理していくものです。

たとえば、**メニュー画面**を次のように構成します。 **メニュー画面**



私たちの作ったプログラムの内容がメニュー画面に見えますね。英単語の追加は

eitan3.bas ———❶
eitan4.bas ———❷

の2つがありますが、改良版の❷だけを使うことにします。

次に、

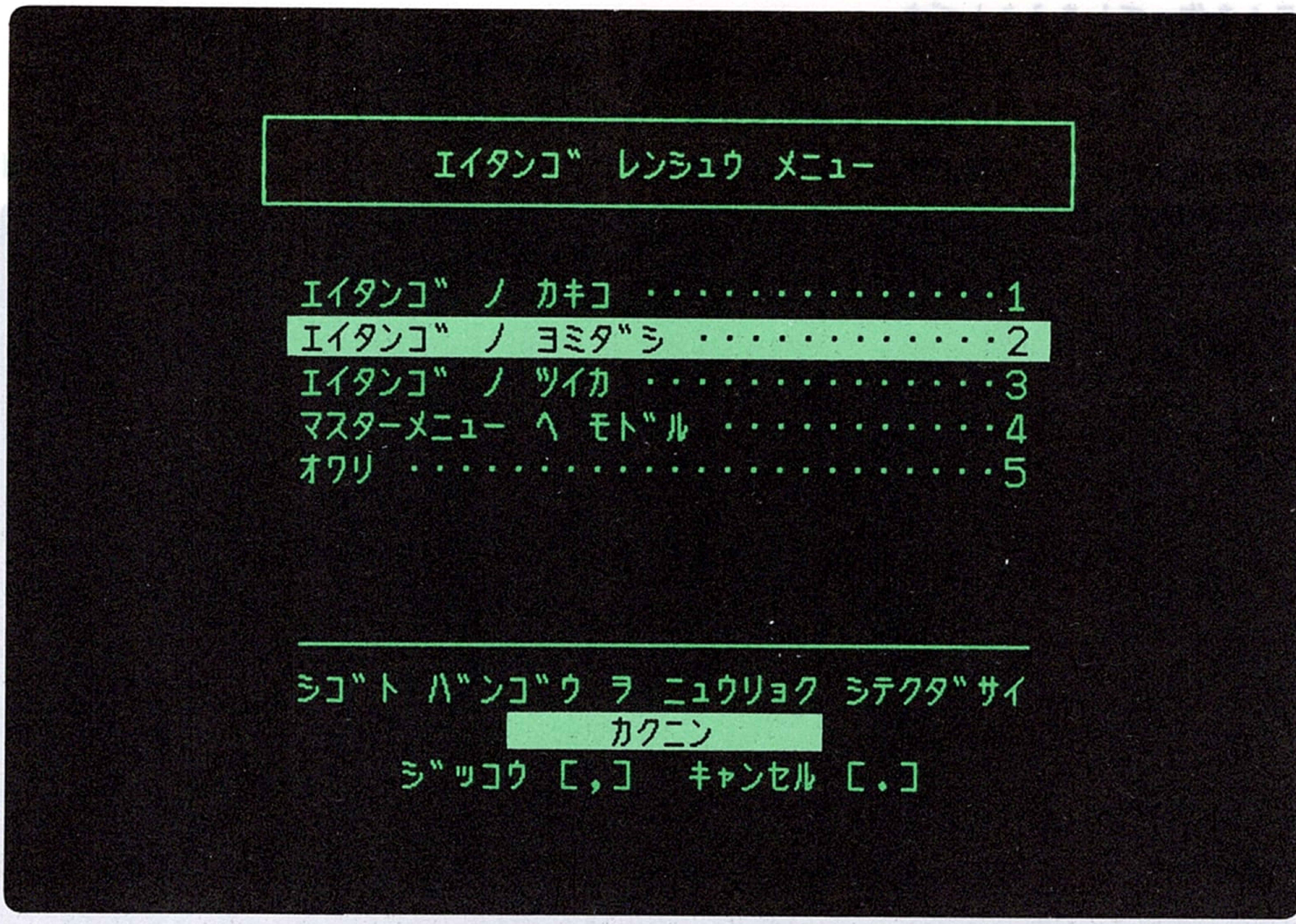
マスターメニューへ・・・・4

というのがありますが、これは、英単語練習メニューとか宛て名書きメニューとかいったメニュープログラムを管理するための、さらにこの上のレベルのメニュープログラムです。これを仮りにマスターメニューとしておきますが、これは本書の最後で作ることになります。

さて上の画面を見ますと、一番下に仕事番号入力をうながすメッセージが表示されていますから、これに従って好きな番号をキーインし

メニューの選択

ます。すると画面は次のように変化します。



反転表示

ブリンク

この画面は2番のメニューを選択したところですが、番号を入力するとリターンキーを押さなくても選択したメニューの部分が**反転表示**されます。

そして**「確認」**という文字が**反転ブリンク**し、その下に、いま選択したメニューを実行するのか、キャンセルするのかを問うメッセージが表示されます。これに対しては、

実行の場合……テンキー側の [,] (コンマ) キー
キャンセルの場合……テンキー側の [.] (小数点) キー

で答えてください。

ここで [,] キーを押して実行を選択すると、**1番のメニュー**なら

```
run "eitan1.bas"
```

2番のメニューなら

```
run "eitan2.bas"
```

3番のメニューなら

```
run "eitan4.bas"
```

4番のメニューなら

```
run "MASTER"
```

が実行されます。

初期設定

それではさっそく**コーディング**（プログラムを組み立てていくことをいいます）していきますが、まず入口の部分からです。

コーディング

```
100 ' file name ···> menu01
110 ' 1985.2.15
120 ' ｴｲﾀﾝｺﾞ ﾚﾝｼｭｳ ﾒﾆｭｰ ﾌﾟﾛｸﾞﾗﾑ
130 ' save "menu01"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
```

これは決まりきったことですから問題ないでしょう。

前ページの画面を見るとおわかりのように、「英単語練習メニュー」という見出しをグラフィックキャラクタで巻いていますが、画面のタテの行数を25行にしておくとタテ罫の間にスキマがあいてしまいます。そこで今回は、WIDTH文の第2パラメータ、CONSOLE文の第2パラメータを25にしておきます。

メニューを配列に読み込む

画面に表示するメニューは**DATA（データ）文**にします。これは入門編で勉強しましたからみなさんごぞんじだと思います。

DATA文

メニューを**READ（リード）文**で配列に読み込み、それをPRINT文で画面に表示すればよいわけです。

READ文

まずDATA文の部分のプログラムを次ページに示します。

610行の横棒は GRPH キーを押しながらテンキー側の ＊ キーを40回たたきます。

また、メニュー名と最後部の番号との間にある「・」（**ナカグロ**）は

ナカグロ

```
610 '─────────────────────────────────
620 '  menu data
630 '─────────────────────────────────
640 DATA 5
650 DATA ｴｲﾀﾝｺﾞ ﾉ ｶｷｺ ···············1,eitan1.bas
660 DATA ｴｲﾀﾝｺﾞ ﾉ ﾖﾐﾀﾞｼ ·············2,eitan2.bas
670 DATA ｴｲﾀﾝｺﾞ ﾉ ﾂｲｶ ···············3,eitan4.bas
680 DATA ﾏｽﾀｰﾒﾆｭｰ ﾍ ﾓﾄﾞﾙ ············4,MASTER
690 DATA ｵﾜﾘ ························5,dummy
```

小数点とかピリオドではありませんよ。これは「ナカグロ」と呼ばれています。

690行のdummyは読んで字のごとく、まったくのダミーです。**他のデータと並びを合わせるため**です。

もしこれを入れておかないとREAD文で読んだ場合に、

Out of DATA

のエラーになると思います。

つまりデータ数がたりないということです。

もう1つデータ文でよく間違えるのは、データ並びはコンマで区切っていますが、コンマをつけ忘れてしまうことです。このときも、コンマで区切るべき2つのデータが1つのデータとして読まれてしまうために、やはりデータ不足のエラーになります。

あっ、そうそう、データ文のミスにもう1つありました。

それは「DATA」というつづりを間違えて「DATE」としてしまうことです。

この場合には直接エラーメッセージは表示されないかもしれませんが、正しい結果を得ることができません。

メニューの読み込み

次はメニューを読む部分です。

```
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
```

最初の

READ MENU

READ文

で、640行の「5」を読みとります。これは**メニューの数**を示しています。メニューを追加するごとにこの数値も変更してください。

次に、このMENUという変数に代入された数を使って**配列を宣言**します。配列は**DIM文**によって宣言します。

配列の宣言

DIM文

ここで紹介するプログラムはメニューが8つしか入らないものなので、当然メニュー数は10個以内ですから**配列宣言は必要ない**のですが、クセをつけるということと、少ない配列でも一応宣言しておいたほうが、わずかではありますがメモリの節約になりますので入れておきました。

●DIMは「ディム」または「ディメンション」と読みます。

なお、**MENU**および**MENU$**という同じ変数名を使っていますが、前者は**数値変数**、後者は**文字型（ストリング）変数**〈$マークがついています〉ですから**まったく違った変数**として扱われます。

数値変数

ストリング変数

メニュー画面の表示

さて次はメニュー画面を作ります。

まずプログラムを示します。

```
230 '
240 PRINT TAB(20)
250 PRINT"┌────────────────────────────────┐"
260 PRINT TAB(20)
270 PRINT"|      ｴｲﾀﾝｺﾞ ﾚﾝｼｭｳ ﾒﾆｭｰ      |"
280 PRINT TAB(20)
290 PRINT"└────────────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
```

|←──── 32字分 ────→|

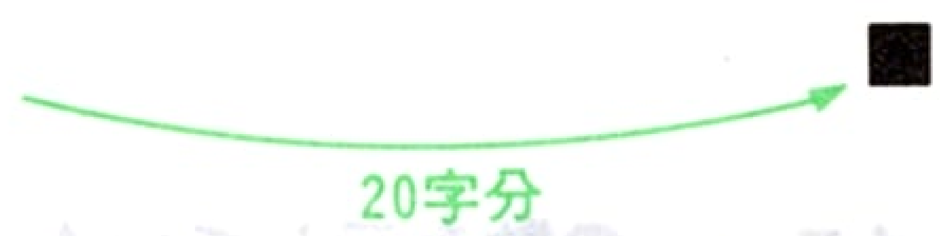
240行の**TAB(タブ)関数**はわかりますね。

TAB関数

画面の左端からカッコ内の数値に相当するキャラクタ分だけ飛ばしてカーソルをその次へもっていきます。上の例ですと、

PRINT TAB(20)

を実行することによってカーソルは次のように移動します。



また、メニュー表示のY座標位置は、タイトルの下を1行開けて5行目からになりますが、座標は0から始まりますので実際には

Y=4

のところから表示することになります。そしてここから1ずつ下へずらしてメニューを表示していけばよいので、結局ループ変数の値をつかって

LOCATE(ロケート)

LOCATE 22, I+3

とすればよいのです。

ジョブナンバーの入力

次は仕事番号入力までです。プログラムは次のとおりです。

```
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"ー")  ←横の直線を引く
380 LOCATE 22,13
390 PRINT "ｼｺﾞﾄ ﾊﾞﾝｺﾞｳ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)  ←文字を数値に変換
420 IF N>MENU THEN BEEP : GOTO 400
```

カーソル位置の設定

STRING$関数

360行のLOCATE文で座標(22, 12)のところに**カーソル位置が設定**されますから、そこから**STRING$(ストリングドル)関数**で直線を引きます。STRING$は2通りの使い方があります。

```
string$(28,"ー")
            ↑
            └──文字または数値
string$(28,149)
        ↑
        └──出力する個数
```

カッコ内の第2パラメータで与えられたキャラクタを、第1パラメータで与えられた数だけ出力しますが、第2パラメータは文字でも数値でもよく、**文字の場合**にはダブルコーテーションでくくり、**数値の場合**はキャラクタに対する10進数のアスキーコードを使います。

キャラクタコード

たとえば、グラフィックキャラクタ「ー」のキャラクタコードは下の表から16進数の95ですから、これを10進数に換算すると

9×16+5=149

となります。ですから第2パラメータにこれを使えばよいのですが、こんな計算はめんどうだという人は、16進数の95の前に「**&H**」をつけて、次のようにしても同じことです。

&H

```
print  string$ (28, &H95)
```

キャラクタコード表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	SP	0	@	P		p			SP	ー	タ	ミ		
1	SH	D1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	SX	D2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	EX	D3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	EQ	NK	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	AK	SN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	BL	EB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	BS	CN	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	LF	SB	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	HM	EC	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	CL	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	CR	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	SO	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	SI	↓	/	?	O	_	o	DEL			ッ	ソ	マ	゜		DEL

キー入力待ち

400行のステートメントは**キー入力待ち**ですが、N$＝""のところは、**ダブルコーテーションの間に何も入れない**でください。つまり、["] キーを続けて2回たたくのです。よく、この間にスペースを1つ入れたりしてプログラムが正常に動作しないという問い合わせがあります。

VAL関数

410行は**VAL（バリュー）関数**による、文字の数値への変換です。

INKEY$によるキー入力は文字型変数に代入されますから、これを数値に変換しておいて、あとでメニューの画面出力に使います。

メニュー数より大きな数が指定されていたらキャンセルします。これは420行で行っています。

実行かキャンセルか

いよいよ最後の部分です。

ここでは選択されたメニューを反転表示させ（つまり白地のところにスミ文字で表示されます）、実行するのかキャンセルするのか確認をうながします。

```
430 '
440 COLOR 4
450 LOCATE 22,N+3
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "    ｶｸﾆﾝ    "
510 COLOR 0 : LOCATE 26,15
520 PRINT "ｼﾞｯｺｳ [,]  ｷｬﾝｾﾙ [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
```

COLOR文

現在私たちはモノクロモードでプログラムの作成を行っていますので、**COLOR（カラー）文**はあまり使いませんが、ある部分を強調したいときには白黒モードでのCOLOR文を使うと便利です。

COLOR文の型は

COLOR A

ですが、パラメータAに0～7の値を与えることにより、次の表に示すような機能が利用できます。

A	機　能
0	ノーマル（通常の表示）
1	シークレット（文字は表示されない）
2	ブリンク（点滅する）
3	シークレット（1と同じ）
4	リバース（反転する）
5	リバースシークレット（反転して文字は表示されない）
6	リバースブリンク（反転して点滅する）
7	リバースシークレット（5と同じ）

白黒モードでの
COLOR文

440行ではパラメータに4を指定していますから、これ以降画面に表示されるキャラクタはすべて反転されます。

選択したメニュー番号は変数Nに入っていますから、これを添字とする配列の内容をPRINT文で出力すれば、74ページの画面のようになります。

一度反転表示の用事がすんだら、COLOR文のパラメータに0を指定して通常のモードになおしておきます。これは470行でやっています。480行は**ブザー音**ですね。

500行のステートメントが反転ブリンクの表示です。パラメータに6を指定します。この場合も次の処理に移るときに通常のモードに戻しておきましょう。

530行のキー入力待ちのステートメントに対して何かキー入力されると次のように分岐します。

❶ 押されたのが [,] キーであり、かつ選択された番号がメニュー数に等しければ画面クリアして終りです。

❷ 単に [,] が押されただけならプログラムの実行です。

❸ [,] でもなく [.] でもなければキー入力ミスですから530行へ戻

します。

❹　キャンセルの場合には、現在反転しているメニューを正常モードに戻し、また確認メッセージも消去して、もう一度メニュー選択のやりなおしです。

さあ、これでメニュープログラムが完成しました。まず**プログラムをセーブ**しておきましょう。

```
save "menu01" ↵
```

プログラムの実行

セーブできましたら実行してみましょう。
次の動作テストを行ってください。

❶　メニュー番号5を選び正常に終了するか調べる。
❷　キャンセルキーを押し実際にキャンセルされるかどうか調べる。
❸　メニュー番号2を選び、データの読み出しを行う。

なお、**メニュー番号1は初回のデータ書き込み**ですから、これを選択して実行すると、前のデータは壊されてしまいますので注意してください。

最後にプログラムリストを示しておきます。

```
100 ' file name ····> menu01
110 ' 1985.2.15
120 ' エイタンゴ レンシュウ メニュー プログラム
130 ' save "menu01"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
230 '
240 PRINT TAB(20)
250 PRINT"┌──────────────────────────┐"
260 PRINT TAB(20)
```

```
270 PRINT"|       ｴｲﾀﾝｺﾞ ﾚﾝｼｭｳ ﾒﾆｭｰ       |"
280 PRINT TAB(20)
290 PRINT"└──────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"─")
380 LOCATE 22,13
390 PRINT "ｼｺﾞﾄ ﾊﾞﾝｺﾞｳ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)
420 IF N>MENU THEN BEEP : GOTO 400
430 '
440 COLOR 4
450 LOCATE 22,N+3
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "     ｶｸﾆﾝ     "
510 COLOR 0 : LOCATE 26,15
520 PRINT "ｼﾞｯｺｳ [,]  ｷｬﾝｾﾙ [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
610 '────────────────────────────────────────
620 '  menu data
630 '────────────────────────────────────────
640 DATA 5
650 DATA ｴｲﾀﾝｺﾞ ﾉ ｶｷｺ ················1,eitan1.bas
660 DATA ｴｲﾀﾝｺﾞ ﾉ ﾖﾐﾀﾞｼ ··············2,eitan2.bas
670 DATA ｴｲﾀﾝｺﾞ ﾉ ﾂｲｶ ················3,eitan4.bas
680 DATA ﾏｽﾀｰﾒﾆｭｰ ﾍ ﾓﾄﾞﾙ ············4,MASTER
690 DATA ｵﾜﾘ ························5,dummy
```

2・9 メニューへ戻る

私たちがこの章で組んできたプログラムは、たとえば「単語追加」のメニューを選択しますと、追加処理が終るとプログラムはENDになり、追加したデータを読み出したいときは、もう一度メニュープログラムをロードしてRUNさせなければなりません。

何かよい方法はないでしょうか？

あります、あります。**それぞれのプログラムにメニューへ戻るステートメントを折り込んでやればよい**のです。

「書き込み」「読み込み」「追加」の３つのプログラムのENDがある行番号は次のとおりでした。

❶ **eitan1.bas**（書き込み）………350行
❷ **eitan2.bas**（読み出し）………370行
❸ **eitan4.bas**（追加）……………390行

メニューへ戻すには、この辺を少し変更してやればよいのです。

まず、❶からいきましょう。プログラムをロードしてください。

```
load "eitan1.bas" ⏎
```

そして、次のステートメントをキーインしてください。

```
320 PRINT
330 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
340 FOR I=1 TO 3000 : NEXT I
350 RUN "menu01"
```

350行はEND文でしたが、上のプログラムを入力すると消えてしまいます。

入力できましたら次はセーブです。

```
save "eitan1.bas" ⏎
```

これで１つが出来上りです。

セーブするときにファイル名を間違わないようにしてください。

❷と❸のプログラムについても同様に行ってください。

```
load "eitan2.bas" ⏎
```

```
360 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
370 FOR I=1 TO 3000 : NEXT I
380 RUN "menu01"
```

```
save "eitan2.bas" ⏎
```

```
load "eitan4.bas" ⏎
```

```
360 PRINT
370 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
380 FOR I=1 TO 3000 : NEXT I
390 RUN "menu01"
```

```
save "eitan4.bas" ⏎
```

以上の操作が終ったら、メニュープログラムを実行してデータ読み出しかデータ追加を選択し実際にためしてみてください。

そして最後はメニューに戻って終りにしてみましょう。

第3章

シーケンシャルファイル[II]

3・1 メニュープログラム

前章では「英単語練習プログラム」を例にとってシーケンシャルファイルの扱い方について説明してきました。

この章では、**電話帳プログラムを**例にとって、シーケンシャルファイルについて、もう少し勉強しておきましょう。

電話帳プログラムとは、名前を入力するとその人の電話番号を表示してくれるものです。考え方は英単語の場合とまったく同じですが、みなさんにもう少し慣れていただくためと、検索テクニックがありますのでとりあげることにしました。また、プリンタも使ってみることにしましょう。

ここで扱う仕事は次の5つです。

❶ **電話帳ファイルの作成**――このプログラムは新規のファイル作成のときのみ使用します。

❷ **電話番号の追加**

❸ **電話番号の修正**――すでにファイルに書き込んであるデータを修正します。

❹ **電話番号の検索**

❺ **電話番号リストの印刷**

メニュー画面のレイアウトは右ページのようにしました。

これは、前章のメニュープログラムとほぼ同じですので、

menu01

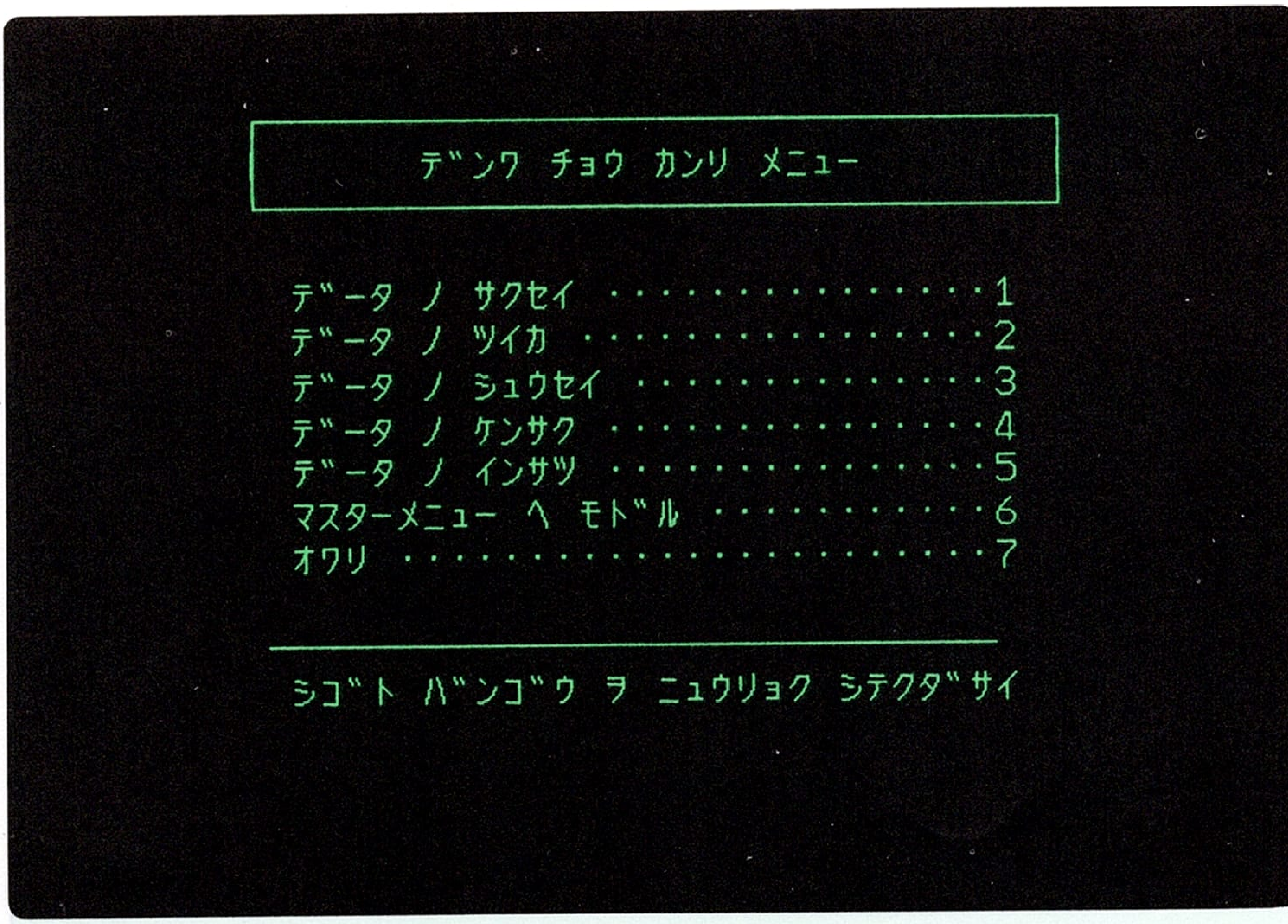
をロードし、修正・追加して使うことにします。

それではまず、前章のメニュープログラムをロードしてください。

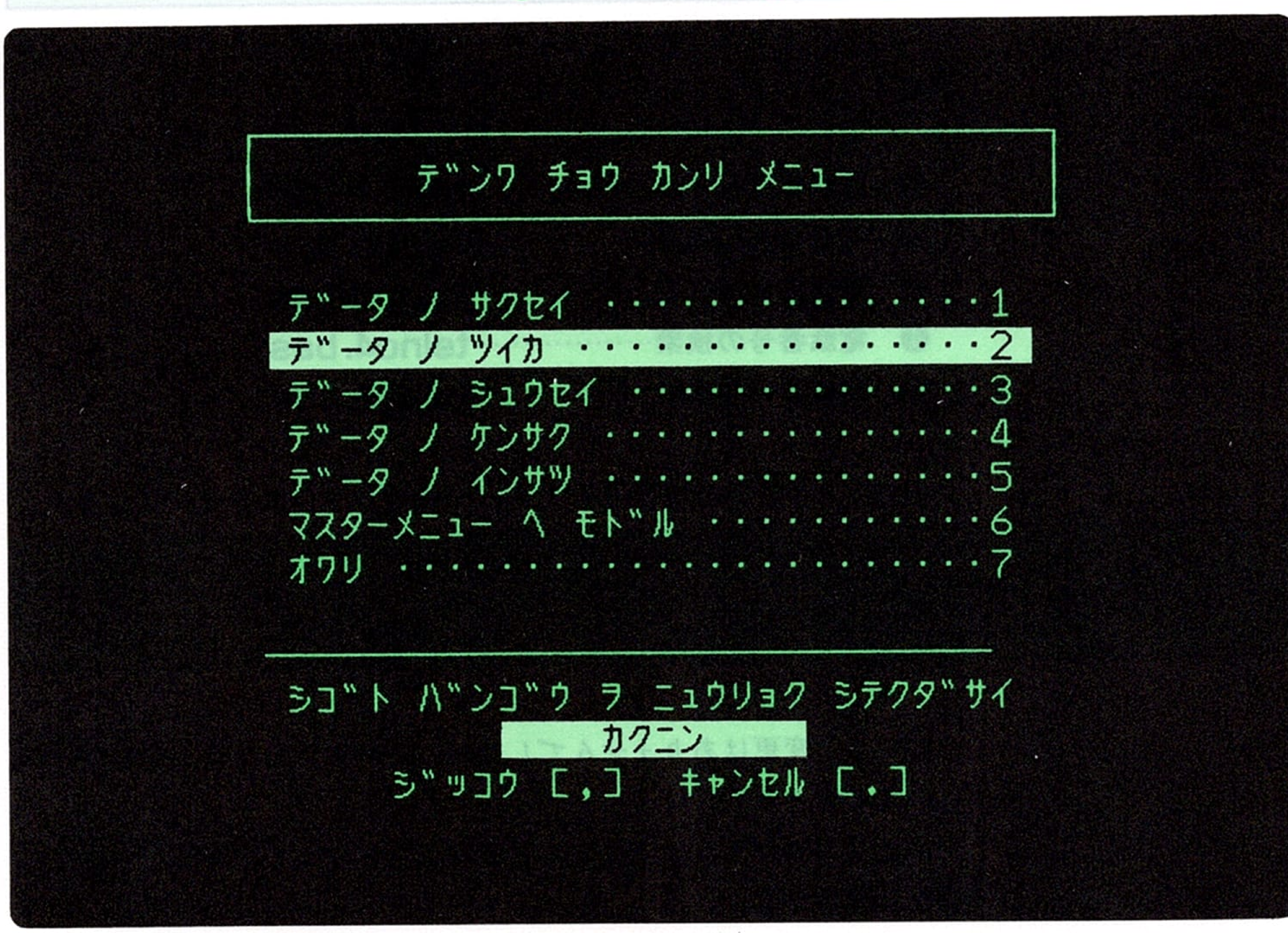
```
load "menu01" ⏎
```

ロードできましたら、修正するところを次に示しますから、 LIST コマンドでその部分を画面に表示し、カーソル移動で修正してください。修正したら必ずリターンキーを押してください。

```
100 ' file name ････> menu02 ←―――ファイル名変更
110 ' 1985.2.15
120 ' デンワ チョウ プログラム ←―――処理内容の変更
130 ' save "menu02" ←―――ファイル名変更
```



●メニュー初期画面●



●メニュー番号を選んだところ●

そして270行の見出しも変更しましょう。

```
270 PRINT"|        ﾃﾞﾝﾜ ﾁｮｳ ｶﾝﾘ ﾒﾆｭｰ        |"
```

610行からの**DATA文**は修正と追加になります。

スクリーンエディタで修正しても、最初からキーインしなおしてもどちらでもかまいません。

```
610 '────────────────────────────
620 ' menu data
630 '────────────────────────────
640 DATA 7
650 DATA ﾃﾞｰﾀ ﾉ ｻｸｾｲ ················1,telno1.bas
660 DATA ﾃﾞｰﾀ ﾉ ﾂｲｶ ·················2,telno2.bas
670 DATA ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ ···············3,telno3.bas
680 DATA ﾃﾞｰﾀ ﾉ ｹﾝｻｸ ················4,telno4.bas
690 DATA ﾃﾞｰﾀ ﾉ ｲﾝｻﾂ ················5,telno5.bas
700 DATA ﾏｽﾀｰﾒﾆｭｰ ﾍ ﾓﾄﾞﾙ ············6,MASTER
710 DATA ｵﾜﾘ ·························7,dummy
```

このリストを見てわかるとおり、これから作成するプログラム名は次のようにしました。

❶ 電話帳ファイルの作成………telno1.bas
❷ 電話番号の追加………………telno2.bas
❸ 電話番号の修正………………telno3.bas
❹ 電話番号の検索………………telno4.bas
❺ 電話番号の印刷………………telno5.bas

上のリストが入力できましたら、**このプログラムをセーブ**します。

```
save "menu02"
```

たいした変更はありませんでしたが、プログラムリストを右ページにのせておきます。

使えるプログラムは流用すると便利だということがおわかりいただけたと思います。

```
100 ' file name ････> menu012
110 ' 1985.2.15
120 ' ﾃﾞﾝﾜ ﾁｮｳ ﾌﾟﾛｸﾞﾗﾑ
130 ' save "menu02"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
230 '
240 PRINT TAB(20)
250 PRINT"┌──────────────────────────────┐"
260 PRINT TAB(20)
270 PRINT"│       ﾃﾞﾝﾜ ﾁｮｳ ｶﾝﾘ ﾒﾆｭｰ        │"
280 PRINT TAB(20)
290 PRINT"└──────────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"-")
380 LOCATE 22,13
390 PRINT "ｼｺﾞﾄ ﾊﾞﾝｺﾞｳ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)
420 IF N>MENU THEN BEEP : GOTO 400
430 '
440 COLOR 4
450 LOCATE 22,N+3
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "    ｶｸﾆﾝ    "
```

```
510 COLOR 0 : LOCATE 26,15
520 PRINT "ｼﾞｯｺｳ [,]  ｷｬﾝｾﾙ [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
610 '──────────────────────────────
620 '  menu data
630 '──────────────────────────────
640 DATA 7
650 DATA ﾃﾞｰﾀ ﾉ ｻｸｾｲ ················1,telno1.bas
660 DATA ﾃﾞｰﾀ ﾉ ﾂｲｶ ·················2,telno2.bas
670 DATA ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ ···············3,telno3.bas
680 DATA ﾃﾞｰﾀ ﾉ ｹﾝｻｸ ················4,telno4.bas
690 DATA ﾃﾞｰﾀ ﾉ ｲﾝｻﾂ ················5,telno5.bas
700 DATA ﾏｽﾀｰﾒﾆｭｰ ﾍ ﾓﾄﾞﾙ ···········6,MASTER
710 DATA ｵﾜﾘ ························7,dummy
```

3·2 電話帳ファイルの作成

電話帳ファイルの作成プログラムは、データを次々に入力し、それをディスクに書き込めばよいわけですから、47ページのプログラムにスクリーンエディタを使って手を加えればできてしまいます。

〈注〉47ページのプログラムには2・9節でメニューへ戻る部分が追加してあります。

まず、プログラムをロードしてください。

```
load "eitan1.bas"
```

修正ずみのプログラムを次にかかげます。

```
100 ' file name ････> telno1.bas
110 ' 1985.2.15
120 ' デ゛ンワ ハ゛ンゴ゛ウ ノ トウロク
130 ' save "telno1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 OPEN "2:telno1.dat" FOR OUTPUT AS #1
180 LOCATE 0,3
190 PRINT "オワリ ノ トキ ハ 999 ヲ ト゛ウゾ゛"
200 LOCATE 0,0
210 INPUT "ナマエ ヲ ト゛ウゾ゛      ";A$
220 IF A$="999" THEN GOTO 290
230 INPUT "  デ゛ンワ ハ゛ンゴ゛ウ ハ";B$
240 WRITE #1,A$,B$
250 C=C+1
260 LOCATE 0,0 : PRINT SPC(50)
270 LOCATE 0,1 : PRINT SPC(50)
280 GOTO 200
290 CLOSE #1
300 CLS
310 PRINT "トウロク ソウスウ";C;"ケン デ゛ス"
320 PRINT
330 PRINT "メニュー ヘ モト゛リマス"
340 FOR I=1 TO 3000 : NEXT I
350 RUN "menu02"
```

- 仕事名の変更（120行）
- プログラム名の変更（100行・130行 telno1.bas）
- データファイル名の変更（170行 2:telno1.dat）
- プロンプト文の変更（210行・230行）
- 「タンゴ」を「ケン」に変更（310行）
- メニュープログラム名の変更（350行 menu02）

アウトプットモードでOPENするデータファイル名、このプログラム名、メニュープログラム名、プロンプト文などの変更をします。

変更したところは、上の画面でコメントを入れてある部分です。あとはすべてそのままです。

まだ実行しなくてけっこうです。データ読み出しの節で合わせて行うことにしましょう。セーブを忘れないように……。

```
save "telno1.bas" ⏎
```

3・3 データの追加プログラム

このプログラムも前章の

eitan4.bas

を手直しして使えそうです。まずプログラムをロードしてください。

```
load "eitan4.bas" ⏎
```

ロードできましたら、前節と同じように修正した箇所を示しますから、スクリーンエディタで変更してみてください。

```
100 ' file name ････> telno2.bas
110 ' 1985.2.15
120 ' ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾉ ﾂｲｶ
130 ' save "telno2.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 GOSUB *SHIGOTO
180 GOSUB *YOMU
190 GOSUB *HYOOJI
200 C=0
210 OPEN "2:telno1.dat" FOR APPEND AS #1
220 LOCATE 0,3
230 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
240 LOCATE 0,0
250 INPUT"ﾅﾏｴ ｦ ﾄﾞｳｿﾞ ";A$
260 IF A$="999" THEN GOTO 330
270 INPUT "  ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳﾊ";B$
272 GOSUB *KAKUNIN
280 WRITE #1,A$,B$
290 C=C+1
300 LOCATE 0,0 : PRINT SPC(50)
310 LOCATE 0,1 : PRINT SPC(50)
320 GOTO 240
330 CLOSE #1
340 CLS
350 PRINT "ﾂｲｶ ｿｳｽｳ";C;"ｹﾝ ﾃﾞｽ"
```

- 100, 130: プログラム名の変更
- 120: 処理名の変更
- 210: データファイル名の変更
- 250, 270: プロンプト文の変更
- 350: 「ゴ」を「ケン」に変更

```
360 PRINT
370 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
380 FOR I=1 TO 3000 : NEXT I
390 RUN "menu02"
1000 ' ───────────────────────────
1010 *SHIGOTO
1020 ' ───────────────────────────
1030 PRINT "ﾃﾞｰﾀ ﾉ ﾂｲｶ ｦ ｼﾏｽ"
1040 FOR I=1 TO 3000 : NEXT I
1050 RETURN
1060 '
2000 ' ───────────────────────────
2010 *YOMU
2020 ' ───────────────────────────
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   INPUT #1,A$,B$
2060   C=C+1
2070 WEND
2080 CLOSE #1 : RETURN
2090 '
3000 ' ───────────────────────────
3010 *HYOOJI
3020 ' ───────────────────────────
3030 PRINT
3040 PRINT "ﾏﾂﾋﾞ ﾉ ﾅﾏｴ ･･･> ";A$
3050 PRINT "ﾄｳﾛｸ ｿｳｽｳ･････>";C;"ｹﾝ"
3060 PRINT
3070 PRINT "OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
3080 I$=INKEY$ : IF I$="" THEN GOTO 3080
3090 CLS
3100 RETURN
4000 ' ───────────────────────────
4010 *KAKUNIN
4020 ' ───────────────────────────
4030 FOR I=1 TO 200
4040   BEEP 1 : BEEP 0
4050 NEXT I
4060 LOCATE 0,3
4070 PRINT "ﾄｳﾛｸ ｼﾏｽｶ ( y/n ) ?    "
4080 I$=INKEY$ : IF I$="" THEN GOTO 4080
4090 IF I$="Y" OR I$="y" THEN GOTO 4120
```

- 390 "menu02" ← メニュー名の変更
- 2030 "2:telno1.dat" ← データファイル名の変更
- 3040 ナマエ ← 「タンゴ」を「ナマエ」に変更
- 3050 ケン ← 「ゴ」を「ケン」に変更

```
4100 IF I$="N" OR I$="n" THEN GOTO 4120
4110 BEEP : BEEP: BEEP : GOTO 4080
4120 LOCATE 0,3
4130 PRINT "ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 ｦ ﾄﾞｳｿﾞ"
4140 IF I$="Y" OR I$="y" THEN RETURN
4150 RETURN 300
```

これで、「データの追加」部分もできてしまいました。意外と簡単ですね。このように、自分に関連した仕事のプログラムを作る場合、できるだけ同じようなプログラムを修正・追加して流用すると、**バグ**も少なくてすみますし、手間もはぶけます。

バグ
●プログラムの間違いのことをバグといいます。バグは英語で「虫」という意味で、虫とりをすること、つまりプログラムの間違いをなおすことをデバッギングといいます。

スクリーンエディタでの修正方法はわかりますね。たとえば、100行のプログラム名を、

スクリーンエディタ

eitan4.bas——telno2.bas

のように変更したいのであれば、カーソルを**e**の上に重ね、

t e l n o 2

とキーインします。「.」以降は修正の必要がありませんから、ここでリターンキーを押します。これでOKです。

修正が終ったらセーブしておきましょう。

```
save "telno2.bas" ⏎
```

3・4　登録・追加プログラムの実行

この節では、すでに作成ずみの**電話番号書き込み用・追加用のプログラムファイルを実行**してテストをしてみます。また、この結果作られたデータファイル**「telno1.dat」**は次節以降で使うことにいたしましょう。

まず、ドライブ1のディスクにはメニュープログラムが保存されていますから、これをロードして実行するところから始めます。

メニューのロード

```
run "menu02" ⏎
```

89ページのようなディスプレイ画面になりましたか？

いま私たちはデータの書き込みをしようとしているわけですから、仕事番号は「1」をキーインします。

すると、入力に対しての確認を求めてきますから、もしよければコンマキー [,] を押します。

すぐにデータ書き込みのプログラム「teleno1.bas」がロードされ、続いて実行されます。画面がクリアされてデータ入力をうながすメッセージが表示されますから、**テストデータを入力**してみましょう。

次の4名について入力してみましょう。

入力データ

トクガワ　イエヤス……000−000−0000
トヨトミ　ヒデヨシ……111−111−1111
オダ　ノブナガ…………222−222−2222
アケチ　ミツヒデ………333−333−3333

データの入力

次に示す画面は2人目のデータを入力しているところです。

```
ナマエ ヲ ト゛ウソ゛     ? トヨトミ ヒテ゛ヨシ
  テ゛ンワ ハ゛ンコ゛ウ ハ? 111-111-111■

オワリ ノ トキ ハ 999 ヲ ト゛ウソ゛
```

4人の氏名と電話番号を打ち込み、5人目の入力を促すメッセージが表示されたら「999」を入力します。

すると、

トウロク　ソウスウ　4　ケン　デス
メニュー　へ　モドリマス

の表示がでてメニュープログラムが実行されます。

今度は**追加処理メニュー2を選択**してみましょう。1を選択したときと同じ手順で次の画面表示に至ります。このようになっていれば、書き込みは順調にいっているはずです。

```
デ゛ータ ノ ツイカ ヲ シマス

マツヒ゛ ノ ナマエ ・・・> アケチ ミツヒテ゛
トウロク ソウスウ・・・・・> 4 ケン

OK ナラ RET キー ヲ ト゛ウソ゛
```

どうですか？　間違いなくできていましたか？

それでは ↵ キーを押して次へ進みましょう。

以下の操作は2・7節で勉強ずみですので、問題はないと思います。次の6人の電話番号を追加しておきましょう。

サンプルデータの電話番号はきれいにそろっていますが、これは見やすくするためです。何桁でもかまいません。

追加データ

アライ　ハクセキ………444-444-4444
カイバラ　エッケン……555-555-5555
カイホ　セイリョウ……666-666-6666
ヨシダ　ケンコウ………777-777-7777
サトウ　イッサイ………888-888-8888
ヨシダ　ショウイン……999-999-9999

6人のデータ入力が終りましたら「999」を入力し、メニュープログラムに戻ります。

入力データの確認

いま入力したデータ追加の確認をしたい人は、もう一度メニュー番号2を選択していただき、データ追加プログラムを実行してください。登録末尾の氏名、登録総数が表示されますから、確認できましたら、リターンキーではなく **STOP** キーを押して止めてください。

そしてダイレクトモードで「CLOSE」をキーインし、リターンキ

ーを押しておいてください。

```
デ゛ータ ノ ツイカ ヲ シマス

マツヒ゛ ノ ナマエ ・・・> ヨシタ゛ ショウイン
トウロク ソウスウ・・・・・> 10 ケン  ←――確かに登録されていました

OK ナラ RET キー ヲ ト゛ウソ゛
^C  ←――ストップキーを押しました
Break in 3080
Ok
close  ←――万全を期してキーインしておきましょう
Ok
■
```

3·5 データの検索

ここまで進んできたあなたは、もうだいぶシーケンシャルファイルの扱いに慣れたことと思います。私たちが日常かかわっている実務上のファイルと違ってディスクのデータファイルは、いくら書き込まれたといっても、目に見えないものですから実際に読み出して確認してみるまでは不安が残ります。初めのうちはとくに、書き込みがうまくいったのかどうか、追加は確実にできているのかどうか、ほんとうに心配なものです。

しかし、たまには失敗もあるかもしれませんが、実践を積み重ねていくうちにだんだん自信がもてるようになってくると思います。

データの検索

さて、この節では、初めてのテーマ**「データの検索」**をマスターすることにしましょう。

読んで字のごとく、データを探すこと、**自分のほしいデータをコンピュータに探させ画面に表示**させようというわけです。

シーケンシャルファイルとランダムファイルの違いのところで述べましたが、シーケンシャルファイルでは、ある1つのデータだけを直接に読み出すことができません。最初のデータからたんねんに読んでいかなければならないのです。ずいぶん融通がきかないと思いますが、コンピュータがやってくれるのですから、私たちはだまって従うことにいたしましょう。

プログラムの流れとフローチャート

プログラムのコーディングに入る前に、おおまかな流れをつかんでおいたほうがよいと思いますので、簡単に説明しておきます。

データの読み飛ばし

まず、電話帳データファイルをオープンしてINPUTモードで**データの読み飛ばし(カラ読み)**をします。これは、データ総数をチェックし、この数値を添字として、氏名、電話番号格納用の配列を宣言するためです。

検索

次に、キーボードから氏名を入力し、氏名用の配列内に一致するものがあるかどうか**検索**します。

見つからなければ、

トウロク　シテ　アリマセン!!

のエラーメッセージを表示して再度氏名入力に戻ります。

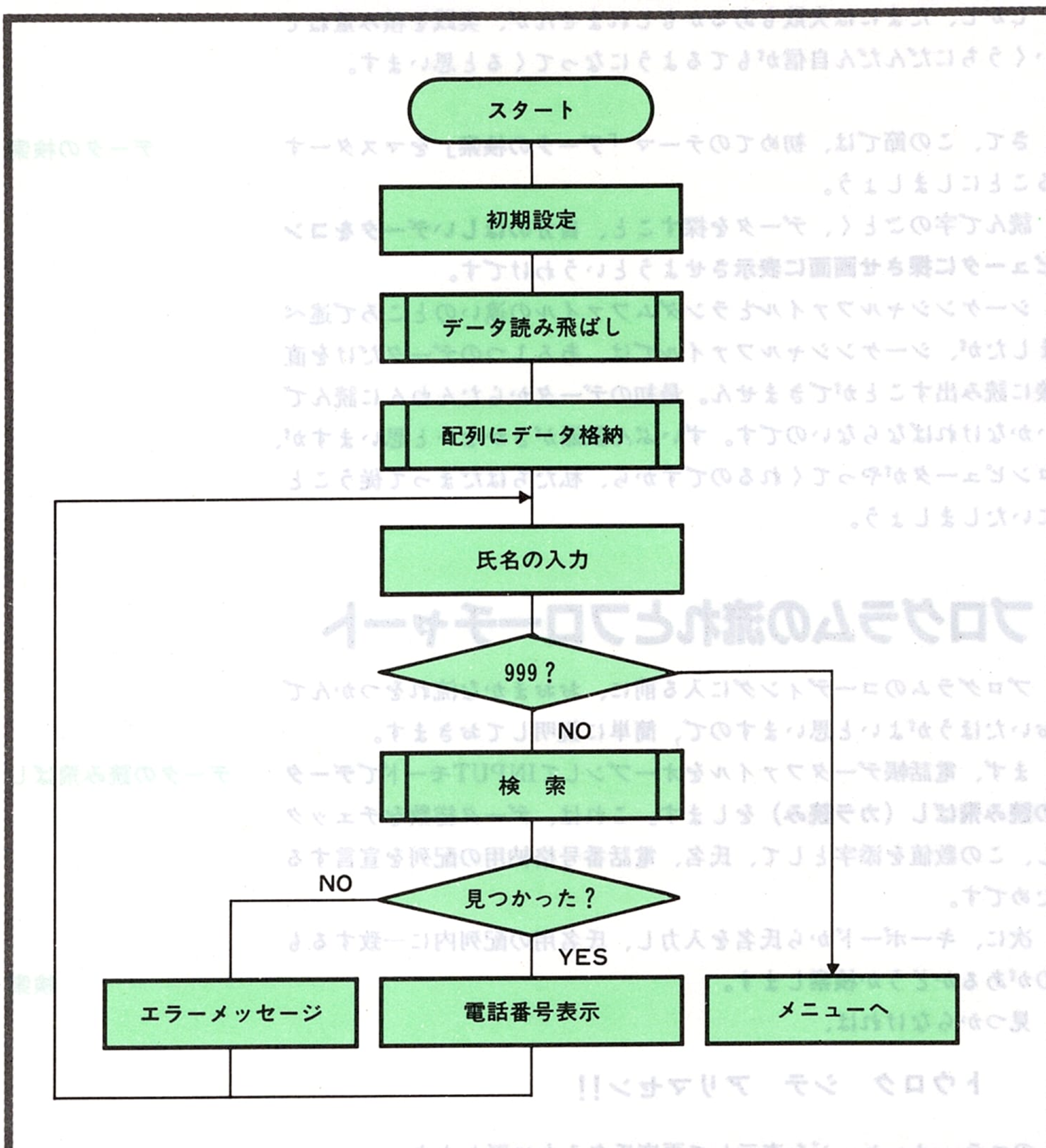
もし見つかれば、その人の電話番号を画面に表示します。そして、確認したらリターンキーを押して次の人の名前を入力します。

フローチャート

●「流れ図」のことで、プログラム処理の流れがよくわかるようにイラスト化したものです。複雑なプログラムでは非常に有効です。

私たちはこれまで**フローチャート（流れ図）**を使いませんでしたが、本節での処理をフローチャートに表わしてみましょう。

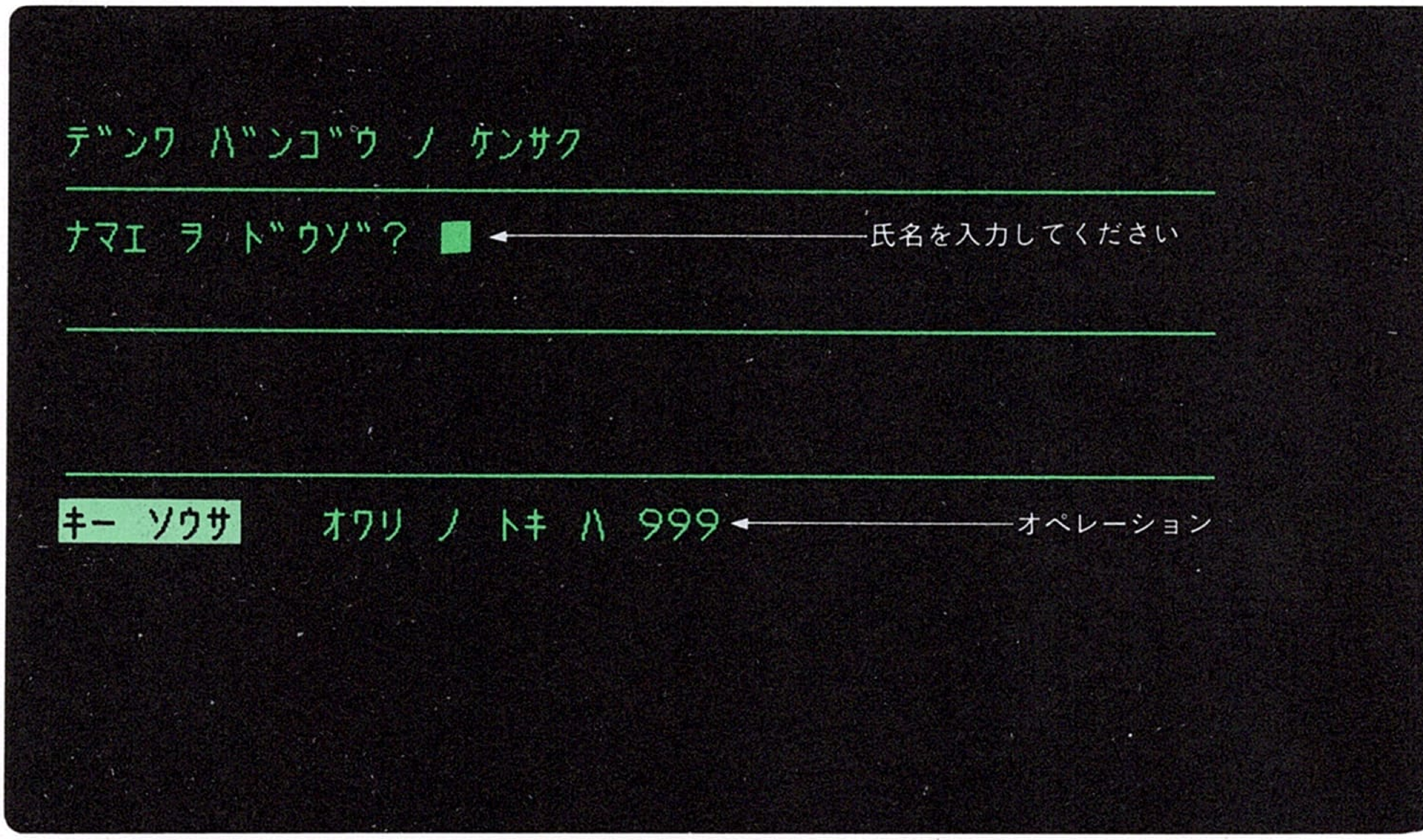
本来は、フローチャートで使う図形は、ディスプレイ表示、条件判断、プリンタ出力など、その処理内容によって形が決められていますが、ここではプログラム処理の流れをつかむだけですから簡略化します。下の図で**菱形は条件判断、両端に2本線のある長方形はサブルーチンへのジャンプ**を表します。



画面レイアウトと操作手順

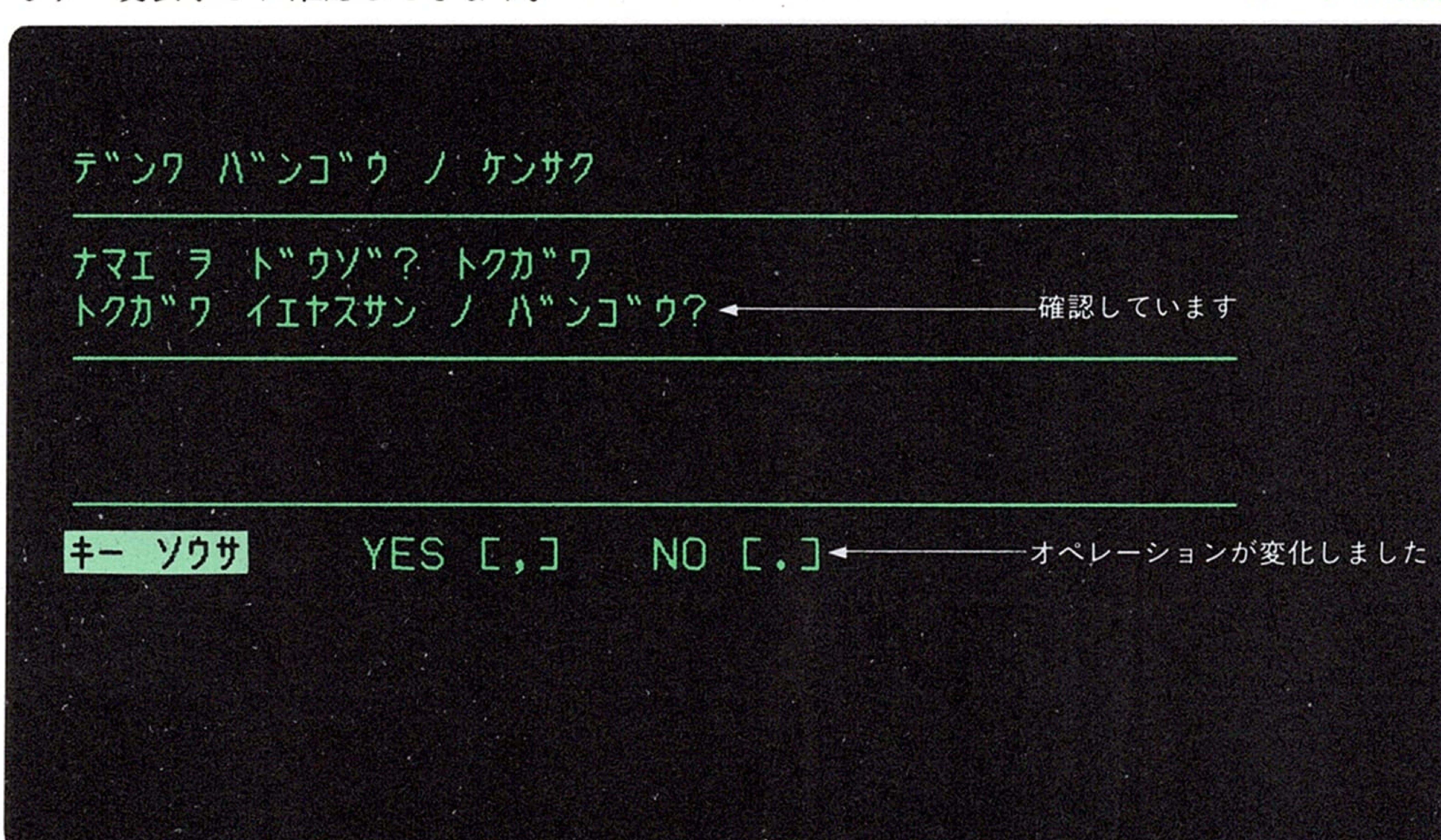
次に、このプログラムを実行した場合、どのように処理が進むのか画面表示を示しながら説明しましょう。前ページのフローチャートも参考にしながら読み進んでください。初期画面は次の通りです。

初期画面



この初期画面で氏名を入力しリターンキーを押すと、**データ検索のサブルーチンへジャンプ**し、もし見つかれば次の画面のように氏名がもう一度表示され確認してきます。

データの確認

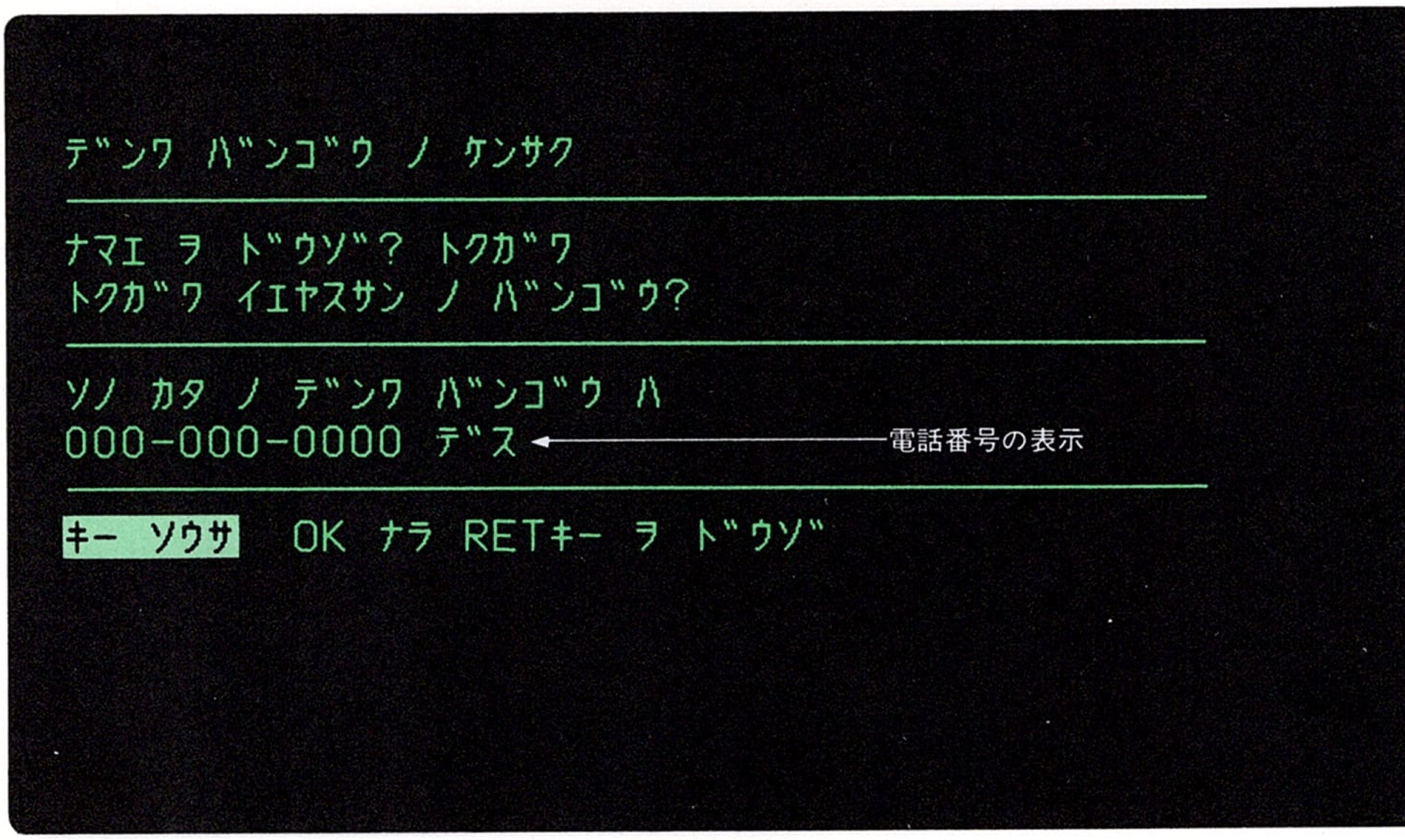


一番下の**キー操作（キーオペレーション）**の表示も場面によって変化することに注意してください。

反転表示
電話番号の表示

キー操作の４文字は目立つように**反転表示**します。モノクロモードで**COLOR文**を使えばよかったですね。

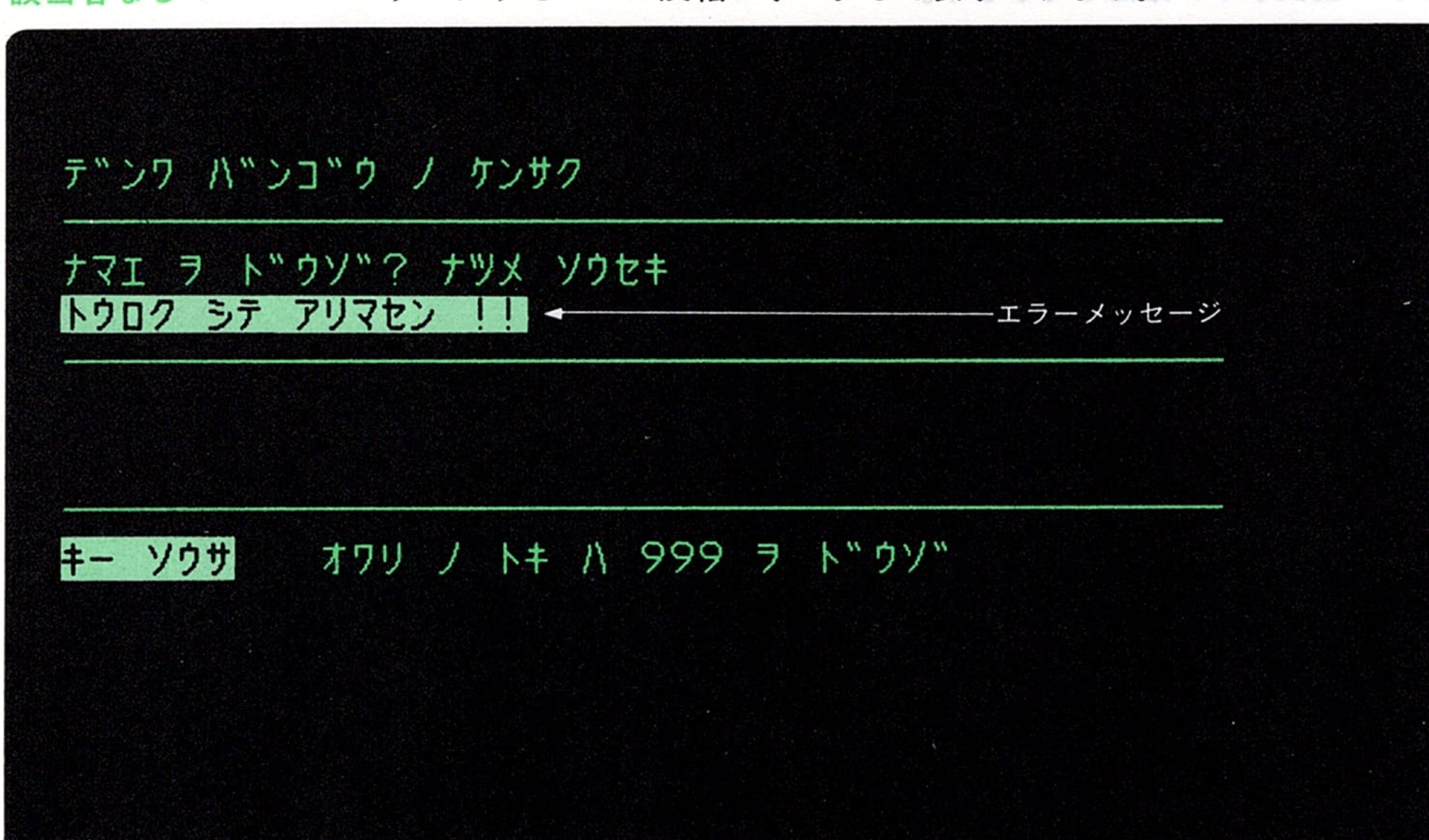
前ページ下の画面でコンマキーを押すと次のようになります。



希望する人の電話番号が表示されましたね。

該当者なし

では該当者がなかったときはどうなるでしょうか？ 次のようにエラーメッセージが**反転ブリンク**して表示されます。



該当者なしの場合、エラーメッセージに対してタイマーがセットしてありますので、2秒くらいで消え、ふたたび初期画面に戻ります。

そしてもし、氏名に対して「999」が入力されると、検索処理は終りとなり、メニュープログラムが実行され、メニュー画面が表示されます。次はメニューへ戻る寸前の画面です。

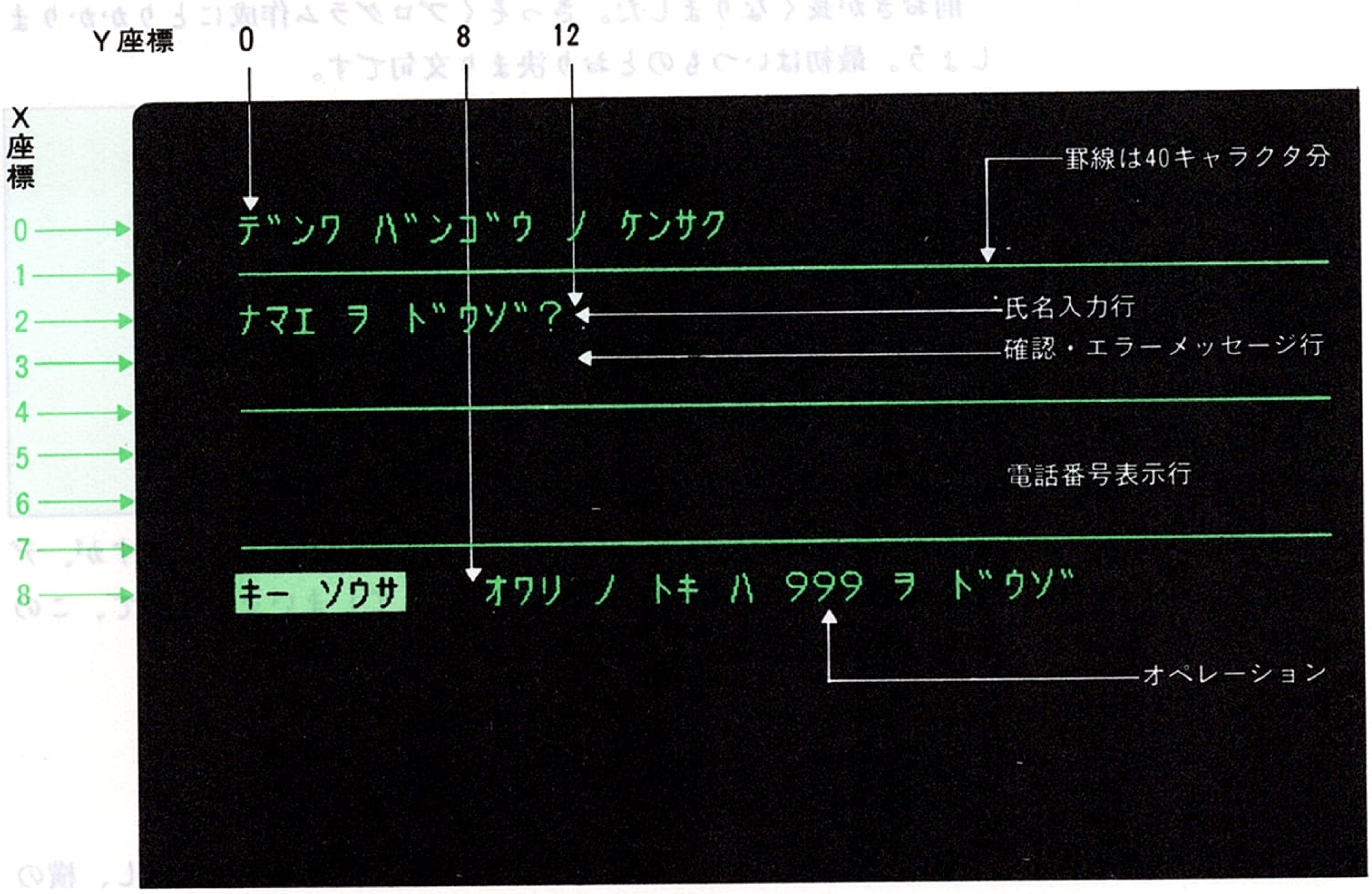
メニューへ戻る

```
メニュー ヘ モト゛リマス
```

画面出力の構成

さて、プログラムに入る前にもう1つ説明しておくことがあります。それは、確認メッセージ、キーオペレーション表示、電話番号表示などの画面出力の際の座標位置です。

次の図をごらんください。



普通、画面を消去するにはCLS命令などを使いますが、これだと画面全体がクリアされてしまい、一部分を消さずに残しておきたいとき不便です。そこで、そのような必要があるときは上のような**画面レイアウト**をあらかじめ作っておき、プログラム中で**LOCATE文**によって

画面レイアウト
LOCATE文

SPC関数

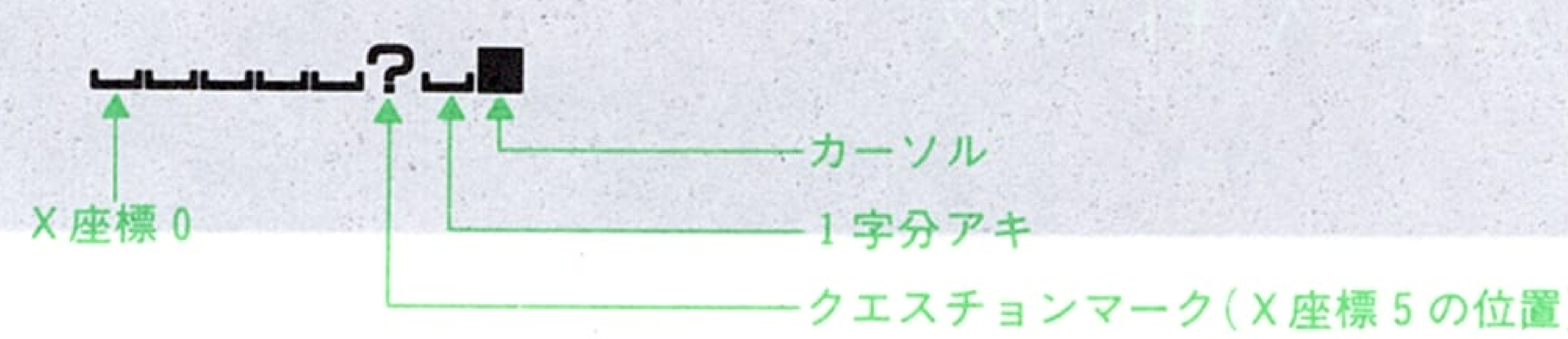
カーソル位置を決め、そこから**SPC関数**でスペースを出力して消去したい文字などを消せばよいでしょう。

INPUT文

なお、**INPUT文**の場合の出力位置ですが、たとえば

LOCATE 5,0 : INPUT A ⏎

を実行した場合、



のように、クエスチョンマークがX座標5の位置に表示され、入力はX座標7の位置からになります。

初期設定

前おきが長くなりました。さっそくプログラム作成にとりかかりましょう。最初はいつものとおり決まり文句です。

```
100 ' file name ····> telno4.bas
110 ' 1985.2.15
120 ' ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾉ ｹﾝｻｸ
130 ' save "telno4.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
```

ファイル名は、これまでの順序でいくと「telno3.bas」ですが、データ検索はメニュー画面で仕事番号4にしてしまいましたので、このようにしました。

枠組みとデータの読み込み

次に、処理名(ジョブ名)、またキー操作などの文字を表示し、横の罫線を3本出力して初期画面の下準備をします。

プログラムを次ページに示します。

白黒反転表示

COLOR文

「キー　ソウサ」という文字は**白黒反転表示**をしますので、出力する前に**COLOR文**でモードを変更しておく必要があります。表示が終ったら通常のモードに戻しておくことを忘れないようにしてください。

```
170 '
180 PRINT "ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾉ ｹﾝｻｸ"
190 PRINT STRING$(40,149)
200 PRINT : PRINT
210 PRINT STRING$(40,149)
220 PRINT : PRINT
230 PRINT STRING$(40,149)
240 COLOR 4 : PRINT "ｷｰ ｿｳｻ"  ←反転表示
250 COLOR 0
260 '
270 GOSUB *YOMU1  ←データ読み飛ばし
280 DIM A$(C),B$(C) : C=0
290 GOSUB *YOMU2  ←配列に格納
300 '
```

また、**STRING＄関数**も前にやりましたからわかりますね。この例の場合、10進のキャラクタコード149（これは16進では95です）を40個出力します。

次は、ドライブ２側のディスクにすでに書き込まれている電話帳データを配列の中に読み込みますが、配列宣言が必要なので最初読み飛ばしをしてデータ数をチェックします。

次のプログラムを入力してください。

カラ読みサブルーチン

```
1000 '————————————————————
1010  *YOMU1
1020 '————————————————————
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
1050   INPUT #1,A$,B$  ←データ読み飛ばし
1060   C=C+1  ←データ数のカウント
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
```

変数Ｃはデータ数を数えるための**カウンタ**です。プログラムを実行したときは変数の内容は０ですから、1050行のINPUT＃文によりデータが読み出されるごとに１ずつ加算されていき、最後には氏名、電話番号を一組とするデータ数がＣの内容となります。

カウンタ

データ末尾のチェックは**EOF文**で行っています。そしてファイル末

EOF文

尾となるまではWHILE～WEND文でループさせデータ読み出しを続けます。

配列の宣言

RETURN文でこのサブルーチンから戻りますと、カウンタCの内容を添字（そえじ）にして、**氏名および電話番号用の配列を宣言**します。これは280行でやっています。

ゼロクリア

そしてもう一度サブルーチンへジャンプして、今度はデータのカラ読みではなく実際に配列内に格納するわけですが、サブルーチンでもう一度カウンタ用変数Cを使いますので**ゼロクリア**(内容を０にする)しておきます。

配列にデータを読む

次のサブルーチンを入力しますが、これは1000行からのサブルーチンとほぼ同じですので**スクリーンエディタ**で行うとよいでしょう。

```
2000 '――――――――――――――――――――――――――――――
2010  *YOMU2
2020 '――――――――――――――――――――――――――――――
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
2080 CLOSE #1
2090 RETURN
2100 '
```

スクリーンエディタ

スクリーンエディタは次の順序で行います。

❶ LISTコマンドで1000～1100行を画面に表示する。
❷ 1000行の「１」にカーソルを合わせ [2] [⏎] 、これで2000行のステートメントができました。
❸ カーソルは1010行の先頭にありますから、やはり [2] [⏎] 。
❹ 同様にして [2] [⏎] を繰り返し1100行まで行う。これで2000行から2100行までのステートメントができました（まだ1000～1100行と内容はまったく同じですが……)。
❺ 次に2010行の＊YOMU1の「１」のところにカーソルを重ね [2] キーを押して [⏎] 。これで正しい2010行ができました。
❻ 現在の行番号2050の「５」を「６」に変更して2060とし、A＄およびB＄の後ろに（C）を挿入してから [⏎] を押してください。
❼ カーソルが次の行頭に移動し、2060行が画面上では２つありますが、現在カーソルがブリンクしている2060行は効力を持っていませ

ん。「6」の上にカーソルを移動し 5 ⏎ としてください。これで新しい2050行ができました。

❽ これでスクリーンエディタの作業は終りです。LISTコマンドで1000〜2100行を画面に表示して確かめてみてください。

2050行のカウンタを添字として配列A＄、B＄の中に氏名、電話番号が格納されていきます。

読み出しが終ったらデータファイルは不要ですので、RETURN文で帰る前に**CLOSE文**を入れて閉じておきましょう。

データ検索と表示

次はメイン部分の入力です。プログラムを示しますので入力してください。

メインルーチン

```
310 *LOOP
320 FOUND=0
330 LOCATE 8,8 : PRINT " ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 "
340 LOCATE 0,2 : INPUT "ﾅﾏｴ ｦ ﾄﾞｳｿﾞ";A$
350 IF A$="999" THEN GOTO 480
360 GOSUB *KENSAKU
370 IF FOUND THEN GOSUB *HYOOJI : GOTO 440
380 BEEP : BEEP
390 LOCATE 0,3 : PRINT SPC(50)
400 LOCATE 0,3 : COLOR 6
410 PRINT "ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !!"
420 FOR I=1 TO 5000 : NEXT I
430 COLOR 0
440 LOCATE 0,3 : PRINT SPC(50)
450 LOCATE 12,2 : PRINT SPC(50)
460 GOTO *LOOP
470 '
```

- 340：氏名入力
- 350：終り
- 360：検索
- 370：該当者あり
- 380〜430：該当者なし
- 440〜450：氏名入力行・エラーメッセージ出力行をクリア

コメントを入れておきましたからプログラムの流れは理解していただけると思います。

LOCATE文のX座標、Y座標の位置は105ページの図を参考にして読み進めてください。どこの行のどの部分を消去するのかよく注意してください。

氏名の入力が「999」でなければ360行に処理が落ちてきますので、入力された名前が登録してあるかどうか検索します。

検索サブルーチン

この部分のプログラムは次のとおりです。

```
3000 '―――――――――――――――――――――
3010  *KENSAKU
3020 '―――――――――――――――――――――
3030 FOR I=1 TO C
3040   IF INSTR(1,A$(I),A$) THEN GOSUB *KAKUNIN
3050   IF FOUND THEN T=I : RETURN
3060 NEXT I
3070 RETURN
3080 '
```

INSTR関数

データの集合体の中から、あるデータを探し出す（検索する）には**INSTR（インストリング）関数**を使います。この関数の書式は次のとおりです。

INSTR（〈数式〉，〈文字列１〉，〈文字列２〉）

〈注〉INSTR関数は文字列の検索に対してのみ有効です。

文字列１の中から**文字列２**を探し、もしあれば、その位置を値として返しますし、なければ０を返します。**数式**は探し始める位置の指定ですが、省略すると**文字列１**の最初から探します。

実際にためしてみるほうが理解が速いでしょう。

文字列**「abcdabcd」**の中から**「bc」**を探す場合を例にして、次のようにダイレクトモードでキーインしてみてください。

```
a$="abcdabcd"
Ok
x=instr(1,a$,"bc") : y=instr(a$,"bc")  ← 数式を指定した場合、省略した場合をテスト
Ok
? x,y
 2             2  ← どちらも同じ結果でした
Ok
z=instr(x+1,a$,"bc") : ? z  ← さらに３文字目から探します
 6  ← 先頭から６文字目にありました
Ok
z=instr(z+1,a$,"bc") : ? z  ← ７文字目から探します
 0  ← これ以上ありません
Ok
■
```

3040行の

INSTR (1, A$(I), A$)

は、氏名用の配列A＄の中から、先に入力された名前（変数A＄に入っている）を探し、**もし見つかれば0でない値を返します〈注〉**から**IF文**の条件が満たされてTHEN文以下を実行します。この場合は確認のためのサブルーチンへ飛ばし、INSTR関数で探し当てた人で間違いないかどうか確かめさせます。

ところで、3040行は

IF INSTR (1, A$(I), A$)〈 〉0 THEN〜

のように**「0でなければ」**という条件を入れておかなければいけないような気もしますが、もともとIF文は、

IF 〈論理式〉 THEN〜

の形をしていて**論理式の値が真なら**（0以外の数なら）THEN文以降を実行してくれますので「＜＞0」をつける必要はありません。

なお、3040行のINSTR関数で、探し始める位置を1に指定してありますが、1の場合には省略してかまいません。

〈注〉氏名で探す場合、たとえば「アライ　ハクセキ」を探すのに「アライ」と入力したとしますと、この文字列は第1文字目からにありますから「1」を返しますし、「ハクセキ」で探したとしますと4文字目からということで「4」を返します。いずれにしても見つかった場合には「0」でない値になります。

IF〜THEN

〈注〉IF文の論理式は0以外の値なら常に「真」です。0のときにのみ「偽」となります。

データの確認

次に、**確認のサブルーチン**です。

```
4000 '―――――――――――――――――――――――――――――
4010  *KAKUNIN
4020 '―――――――――――――――――――――――――――――
4030 LOCATE 8,8
4040 PRINT "  YES [,]   NO [.]"
4050 LOCATE 0,3 : PRINT A$(I);"サン ノ バンゴウ?"
4060 I$=INKEY$ : IF I$="" THEN GOTO 4060
4070 IF I$="," THEN FOUND=-1 : GOTO 4100
4080 IF I$="." THEN GOTO 4100
4090 BEEP : BEEP : GOTO 4060
4100 LOCATE 8,8 : PRINT SPC(50)
4110 RETURN
4120 '
```

探し出した氏名をディスプレイに表示したのち、INKEY＄によるキー入力待ちとなりますが、**もし間違いがなければ** [,] キーが押されますから、

FOUND＝−1

探している人でなければ [.] が押されて、FOUNDは320行で初期設定されたままですから、

FOUND＝0

となります。

このように、このサブルーチンでは確認の結果をFOUNDという変数の中に値としておさめているのです。つまり、このサブルーチンからジャンプ先へ戻ったときに、FOUNDの内容をチェックし、もし、

FOUND＝−1　つまり　**真**

なら、電話番号を知りたい人の名前であるし、もし、

FOUND＝0　つまり　**偽**

なら、違う人ですからキャンセルしなければなりません。

フラグ

〈注〉FOUNDを日本語になおすと「見つけた」という意味で、検索の場合イメージ的にピッタリなので使いました。別にXでもYでもZでも、何でもかまわないのです。

いま、ここで使ったFOUNDのような使い方を**フラグ**といいます。フラグとは英語で「旗」のことです。つまり、何かを判断するための目印にするのです。

さて、この旗（フラグ）を持ってジャンプ先へ戻りますから、今度は3050行に処理が移ります。ここで、

IF FOUND THEN T＝I：RETURN

つまり、もしFOUNDが真なら、そのときのループ変数Iの値（これが氏名の登録順番になっています）を別の変数Tに代入し、メインルーチンへ帰ります。

反対に、もしFOUNDが偽であれば、このIF文にはひっかからず次のNEXT文に落ちていき、検索を続けます。

FOR〜NEXT

検索が一順して、それでも見つからなければ**FOR〜NEXT（フォア〜ネクト）ループ**から抜け出し、FOUND＝0（偽）のままメインルーチンへ帰ることになります。

話がだいぶ遠くへきてしまいましたが、ここではじめてメインルーチンの360行の後ろに戻ります。つまり370行へ処理が落ちていきます

が、**FOUNDが真であった場合**には電話番号表示のサブルーチンへジャンプし、偽の場合（FOUND＝0）には、

トウロク　シテ　アリマセン!!

のエラーメッセージを表示して、ふたたび氏名入力の初期画面に戻ります。

電話番号表示のサブルーチンを次に示します。

```
5000 '―――――――――――――――――――――――――――
5010 '  *HYOOJI
5020 '―――――――――――――――――――――――――――
5030 LOCATE 8,8 : PRINT " OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
5040 LOCATE 0,5 : PRINT "ｿﾉ ｶﾀ ﾉ ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾊ "
5050 PRINT B$(T);" ﾃﾞｽ"
5060 I$=INKEY$ : IF I$="" THEN GOTO 5060
5070 LOCATE 8,8 : PRINT SPC(50)
5080 LOCATE 0,5 : PRINT SPC(50)
5090 LOCATE 0,6 : PRINT SPC(50)
5100 RETURN
```

電話番号は配列B＄に読み込みましたね。ですから、変数Tの内容で示される配列の要素を表示すればよいわけです。

INKEY＄によるキー入力待ちにして、何かキー入力があったら、表示部分をクリアしてリターンです。

氏名に「999」が入力されたらメニューへ戻ります。次のプログラムを入力してください。

```
480 CLS
490 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
500 FOR I=1 TO 5000 : NEXT I
510 RUN "menu02"
520 '
```

そして最後に、忘れずにこのプログラムのセーブです。ファイル名の数字を間違わないようにしてください。**もし「4」を「1」などとするとデータ追加プログラムが消されてしまいます。**

```
save "telno4.bas" ⏎
```

プログラム総リストを次にかかげます。

```
100 ' file name ････> telno4.bas
110 ' 1985.2.15
120 ' デ゛ンワ ハ゛ンコ゛ウ ノ ケンサク
130 ' save "telno4.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 PRINT "デ゛ンワ ハ゛ンコ゛ウ ノ ケンサク"
190 PRINT STRING$(40,149)
200 PRINT : PRINT
210 PRINT STRING$(40,149)
220 PRINT : PRINT
230 PRINT STRING$(40,149)
240 COLOR 4 : PRINT "キー ソウサ"
250 COLOR 0
260 '
270 GOSUB *YOMU1
280 DIM A$(C),B$(C) : C=0
290 GOSUB *YOMU2
300 '
310 *LOOP
320 FOUND=0
330 LOCATE 8,8 : PRINT " オワリ ノ トキ ハ 999 "
340 LOCATE 0,2 : INPUT "ナマエ ヲ ト゛ウソ゛";A$
350 IF A$="999" THEN GOTO 480
360 GOSUB *KENSAKU
370 IF FOUND THEN GOSUB *HYOOJI : GOTO 440
380 BEEP : BEEP
390 LOCATE 0,3 : PRINT SPC(50)
400 LOCATE 0,3 : COLOR 6
410 PRINT "トウロク シテ アリマセン !!"
420 FOR I=1 TO 2000 : NEXT I
430 COLOR 0
440 LOCATE 0,3 : PRINT SPC(50)
450 LOCATE 12,2 : PRINT SPC(50)
460 GOTO *LOOP
470 '
480 CLS
490 PRINT "メニュー ヘ モト゛リマス"
```

```
500 FOR I=1 TO 5000 : NEXT I
510 RUN "menu02"
520 '
1000 '------------------------------------------------
1010  *YOMU1
1020 '------------------------------------------------
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
1050   INPUT #1,A$,B$
1060   C=C+1
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
2000 '------------------------------------------------
2010  *YOMU2
2020 '------------------------------------------------
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
2080 CLOSE #1
2090 RETURN
2100 '
3000 '------------------------------------------------
3010  *KENSAKU
3020 '------------------------------------------------
3030 FOR I=1 TO C
3040   IF INSTR(1,A$(I),A$) THEN GOSUB *KAKUNIN
3050   IF FOUND THEN T=I : RETURN
3060 NEXT I
3070 RETURN
3080 '
4000 '------------------------------------------------
4010  *KAKUNIN
4020 '------------------------------------------------
4030 LOCATE 8,8
4040 PRINT "  YES [,]   NO [.]"
4050 LOCATE 0,3 : PRINT A$(I);"ｻﾝ ﾉ ﾊﾞﾝｺﾞｳ?"
4060 I$=INKEY$ : IF I$="" THEN GOTO 4060
4070 IF I$="," THEN FOUND=-1 : GOTO 4100
4080 IF I$="." THEN GOTO 4100
```

```
4090 BEEP : BEEP : GOTO 4060
4100 LOCATE 8,8 : PRINT SPC(50)
4110 RETURN
4120 '
5000 '────────────────────────────────────────
5010  *HYOOJI
5020 '────────────────────────────────────────
5030 LOCATE 8,8 : PRINT " OK ﾅﾗ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
5040 LOCATE 0,5 : PRINT "ｿﾉ ｶﾀ ﾉ ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾊ "
5050 PRINT B$(T);" ﾃﾞｽ"
5060 I$=INKEY$ : IF I$="" THEN GOTO 5060
5070 LOCATE 8,8 : PRINT SPC(50)
5080 LOCATE 0,5 : PRINT SPC(50)
5090 LOCATE 0,6 : PRINT SPC(50)
5100 RETURN
```

3・6 データの修正

第2章の「英単語練習プログラム」では、シーケンシャルファイルにすでに書き込んだデータの修正は扱いませんでした。

しかし電話番号のファイルとなると、たまには番号変更もあるでしょうし、また姓が変わったりする人もいます。ですから、ある人の名前を入力したら、そのデータを検索して氏名・電話番号ともに、自由に修正することができたら大変便利だと思いませんか？

この章では、この辺を学んでいこうというわけです。

データ修正の手順

シーケンシャルファイルにおけるデータ修正は、まず初めに、現在すでにディスクに書き込まれているデータを、一度すべて読み出し、それを配列にしまっておきます。

そして、ディスプレイ上のメッセージに従って、キーボードから変更したい人の名前を入力します。次に、この入力データをもとに配列の中から探し出し、これをディスプレイに表示させ、正しい氏名、電話番号を入力させ、配列の内容を入れ換えてしまいます。

この操作を修正の必要のあるデータ数だけくりかえし、最後に修正された配列の内容を、データファイルを新規にOPENして書き込みます。これで修正されたマスターファイルの出来上がりです。

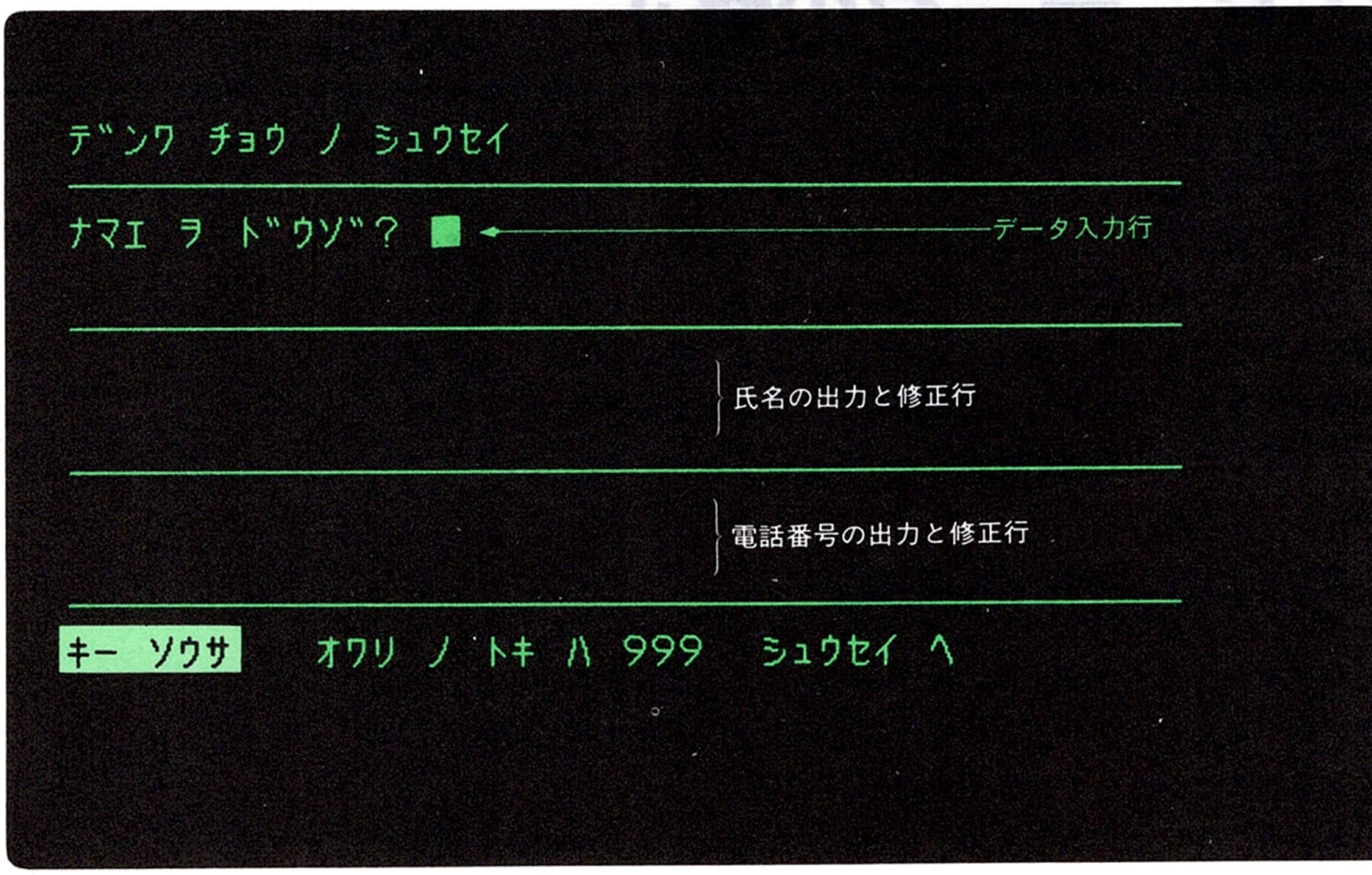
もう、お気づきのことと思いますが、データ読み出しのところ、またプログラムの全体的な流れなど、前節のデータ検索とだいぶ共通しています。そんなわけで、データ検索のプログラムに手を加えたり、あるいは、追加サブルーチンなどを作ったりして完成させていくことにいたしましょう。

みなさんは、まず前節のデータ検索のプログラムをロードしておいてください。そして本節を読みながら、スクリーンエディタで新しいプログラムにしていくことにしましょう。

```
load "telno4.bas" ⏎
```

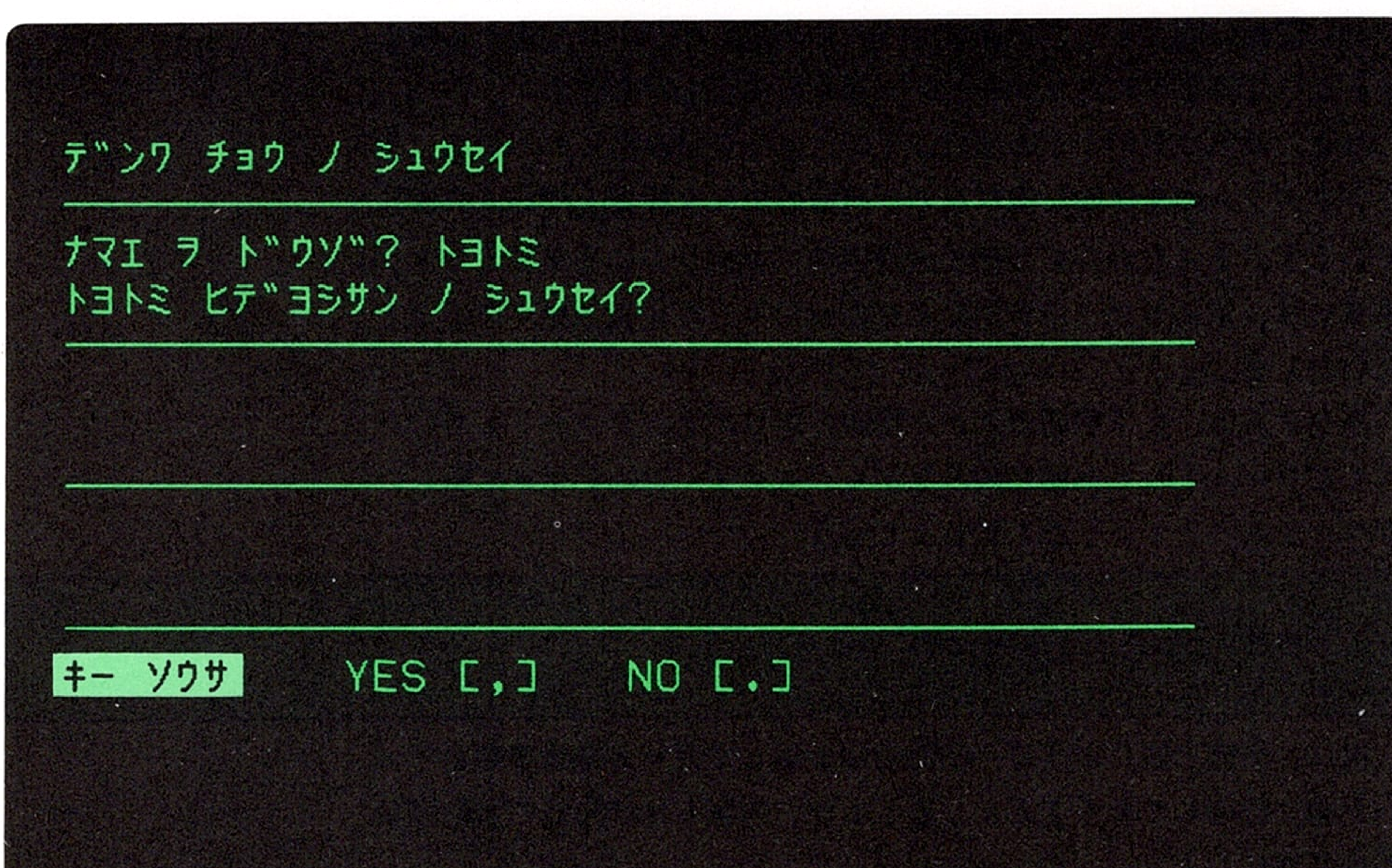
次に、前節と同じように、画面表示を示してデータ修正手順をイメージ的に把握しておくことにしましょう。前回とほぼ同じですので説明は簡単にします。

まず、このプログラムを実行したときの初期画面です。



前回より枠が1つ多くなっています。次からの説明でその意味はわかると思います。

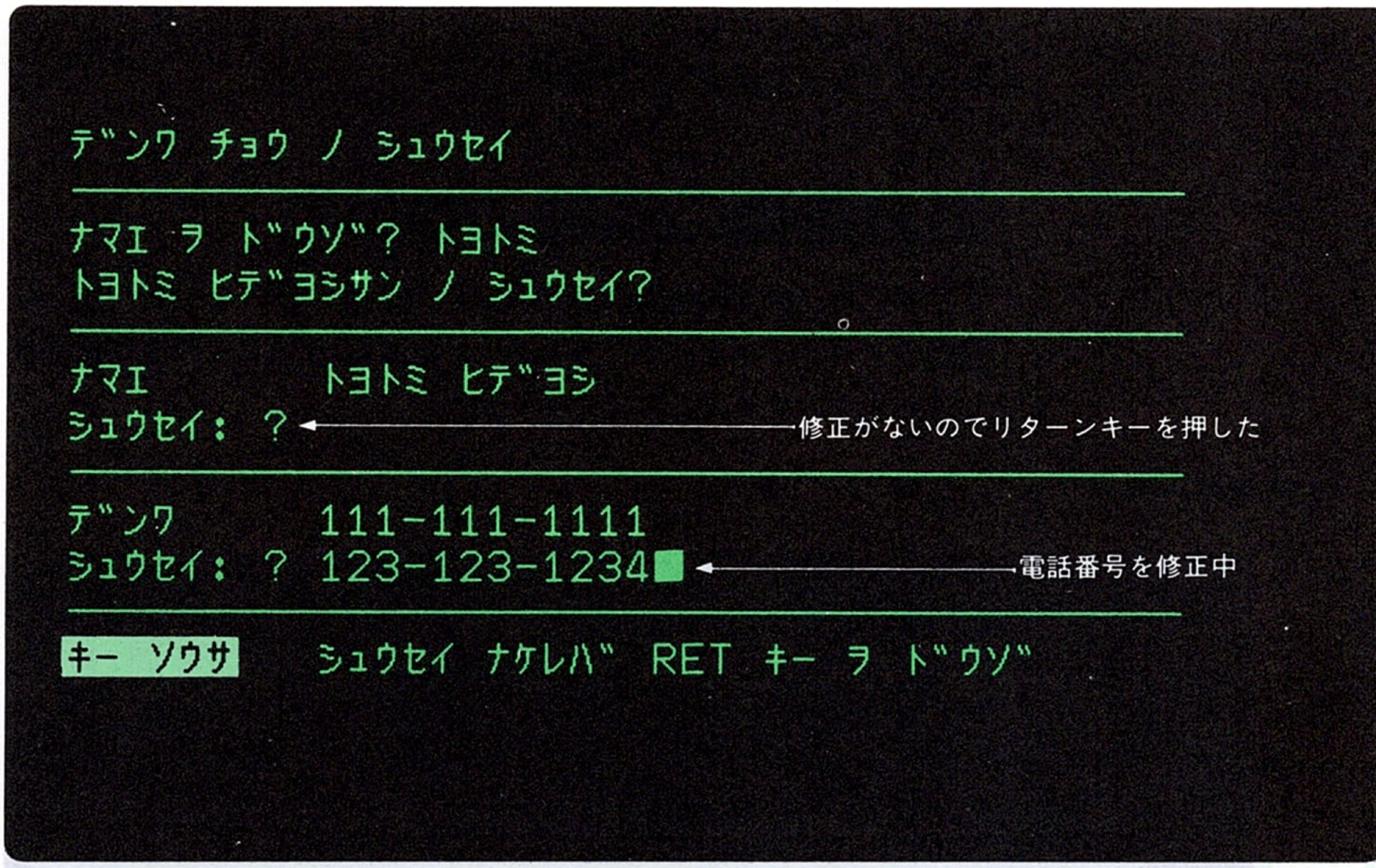
データ入力行に氏名を入れてリターンキーを押すと次のようになります。確認をうながしているのです。



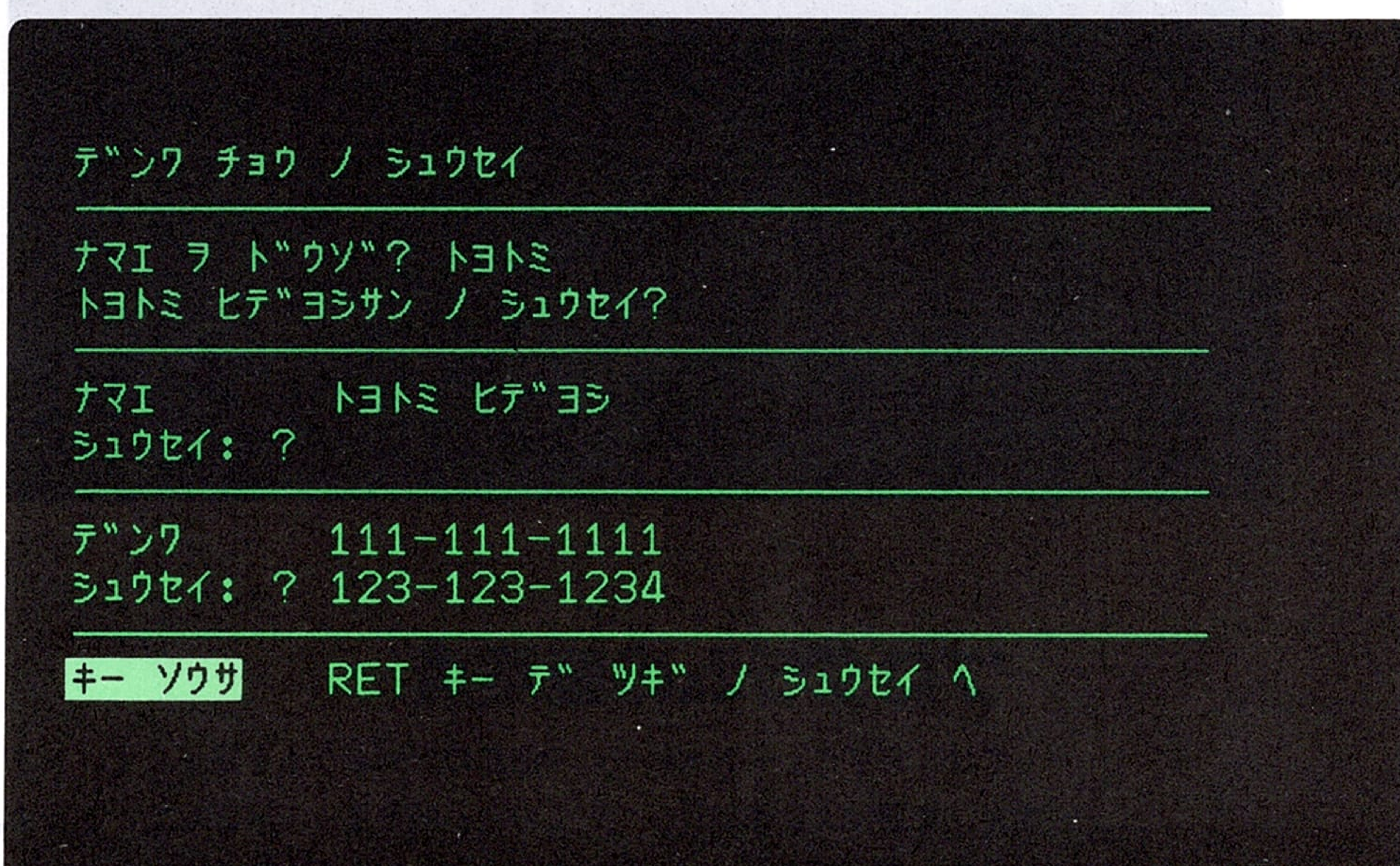
名前はフルネームを入力しなくてもかまいません。修正したい人の

氏名に間違いなかったら [,] キーを押します。すると、修正画面になりますから、もし修正の必要がなければリターンキーを押します。

次の画面は氏名の修正は必要ないのでリターンキーを押し、電話番号の修正を行っているところです。



正しい電話番号を入れてリターンキーを押すと、次のような画面になります。



この画面から キーを押しますと、ふたたび氏名入力画面に戻りますから、もし仕事を終りたければ「999」を入力してください。

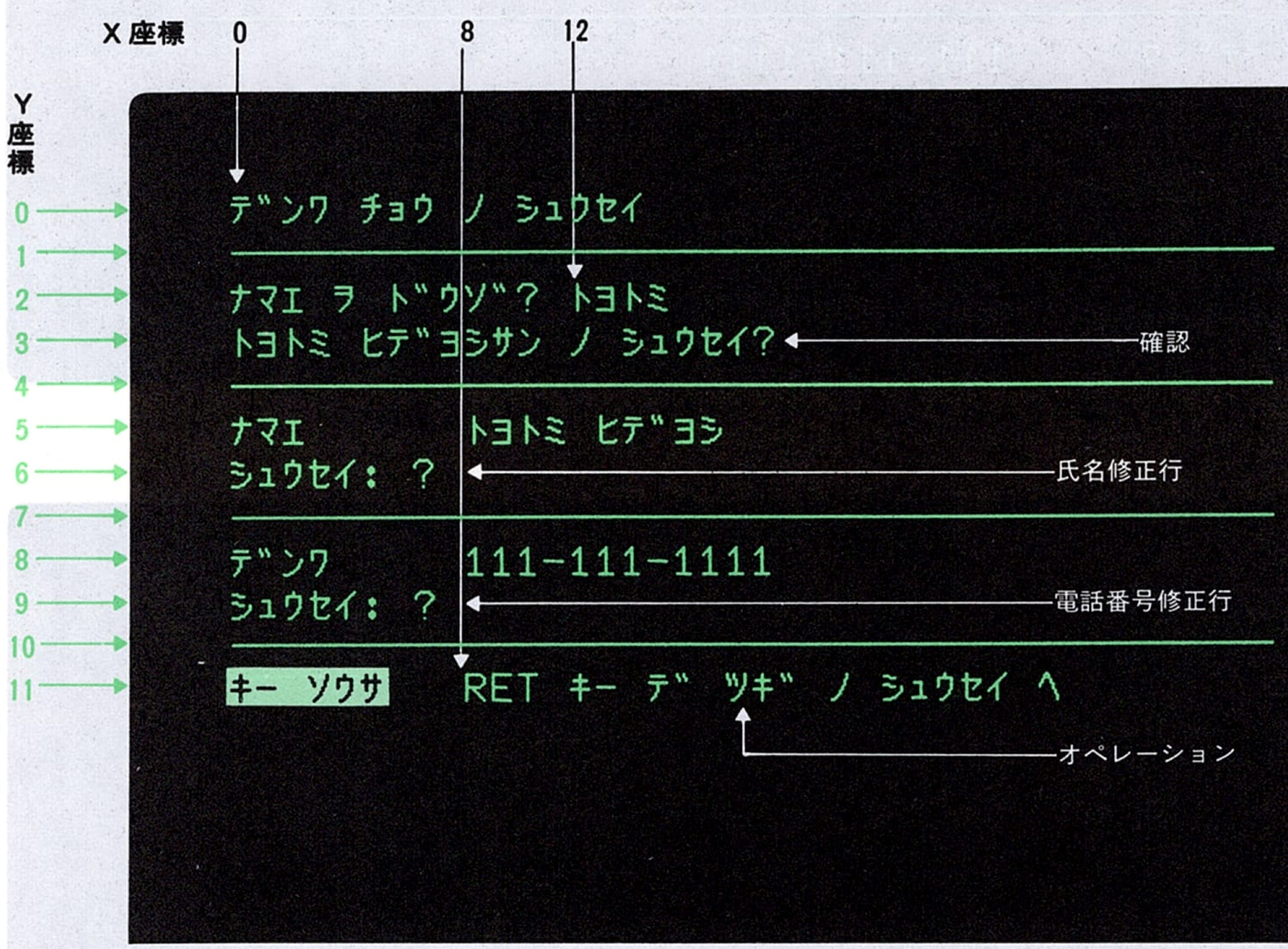
するとデ**ータファイルへの書き込み処理**を行い、修正されたデータが新しいファイルとして登録され、次に**メニュー画面**へ戻ります。

なお、入力した氏名と合致するものがなければ、前節と同じように**エラーメッセージ**が表示されます。

出力画面の座標について

ディスプレイ画面に対しての出力は次のように使います。

前節のプログラムを修正する場合、とくにLOCATE文の座標位置が少し違いますので注意してください。



プログラムの変更点

では、すでにロードされている前節のプログラムを修正していきますが、変更および追加行についてのみ、順を追って解説していきます。みなさんは、スクリーンエディタによるか、あるいは改めて入力する

かしてください。

まず、プログラムヘッドの**REM文の修正**です。

```
100 ' file name ····> telno3.bas
110 ' 1985.2.15
120 ' デ゛ンワ チョウ ノ シュウセイ
130 ' save "telno3.bas"
140 '
```

ファイル名と120行の処理内容を変更してください。ファイル名は

telno4.bas

の「4」を「3」にするだけですからカーソル移動でできますね。

次に、180行の日本語タイトル「電話番号の検索」を次のように変えてください。

```
180 PRINT "デ゛ンワ チョウ ノ シュウセイ"
```

枠が1つふえましたので次の2つのステートメントを追加しておきましょう。

```
232 PRINT : PRINT
235 PRINT STRING$(40,149)
```

330行はLOCATE文のY座標を8から11に変えなければなりません。

```
330 LOCATE 8,11 : PRINT " オワリ ノ トキ ハ 999 "
```

370行は、修正すべきデータであることが間違いなかったときジャンプするサブルーチンは＊HYOOJIではなく**＊SHUUSEI**にして、この部分はまったく新しく作りなおします。とりあえずジャンプ先の**サブルーチン名を変更**しておきましょう。

```
370 IF FOUND THEN GOSUB *SHUUSEI : GOTO 440
```

4000行からのサブルーチンでは、4030行と4100行はLOCATE文のY座標位置、4050行は「バンゴウ？」を「シュウセイ？」になおしておきましょう。いずれもスクリーンエディタで簡単にできます。

```
4030 LOCATE 8,11
```

```
4050 LOCATE 0,3 : PRINT A$(I);"ｻﾝ ﾉ ｼｭｳｾｲ?"
```

```
4100 LOCATE 8,11 : PRINT SPC(50)
```

データ修正サブルーチン

では、データ修正のサブルーチンを完成させてしまいましょう。この部分はだいぶ変わりますので、最初からキーインしてしまったほうが速いかもしれません。

プログラムは右ページに示します。

入力の前に、現在メモリに入っている5000～5100行を

DELETE

●「デリート」と読みます。プログラムの行を削除するときに使います。

DELETE 5000–5100 ⏎

で削除してもよいですが、別にそのままでもかまいません。同じ行番号の場合にはあとから入力されたものが優先され、前のプログラムは消えてしまいますので……。

キーボードからの入力はINPUT文で行います。そしてもしリターンキーが押されれば、その項目は修正しなかったことになりますから、5100行のTHEN文以降でやっているように次の項目表示へジャンプさせてしまいます。

配列の内容の変更

そして、もし修正データが入力されたのであれば、そのデータを配列の中に格納されている古いデータと

A$(T)=A$

の操作により入れ換えています。

名前の修正に対してもまったく同じことを電話番号についても行います。

電話番号の場合には配列Ｂ＄にしまいます。

こうしておいて、あとでまとめてシーケンシャルファイルに書き込むのです。

```
5000 '────────────────────────────
5010  *SHUUSEI
5020 '────────────────────────────
5030 LOCATE 8,11
5040 PRINT " ｼｭｳｾｲ ﾅｹﾚﾊﾞ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
5050 LOCATE 0,5 : PRINT "ﾅﾏｴ       ";A$(T)
5060 PRINT "ｼｭｳｾｲ:"
5070 LOCATE 7,6 : INPUT A$
5080 IF A$="" THEN GOTO 5100
5090 A$(T)=A$
5100 LOCATE 0,8 : PRINT "ﾃﾞﾝﾜ       ";B$(T)
5110 PRINT "ｼｭｳｾｲ:"
5120 LOCATE 7,9 : INPUT B$
5130 IF B$="" THEN GOTO 5150
5140 B$(T)=B$
5150 LOCATE 8,11 : PRINT SPC(50)
5160 LOCATE 8,11
5170 PRINT " RET ｷｰ ﾃﾞ ﾂｷﾞ ﾉ ｼｭｳｾｲ ﾍ"
5180 I$=INKEY$ : IF I$="" THEN GOTO 5180
5190 LOCATE 8,11: PRINT SPC(50)
5200 LOCATE 0,5 : PRINT SPC(50)
5210 LOCATE 0,6 : PRINT SPC(50)
5220 LOCATE 0,8 : PRINT SPC(50)
5230 LOCATE 0,9 : PRINT SPC(50)
5240 RETURN
5250 '
```

なお、このルーチンでデータを修正したつもりが、その修正データさえも間違っていた場合、このプログラムではそのフォローをしていませんので、仕方ないですからそのデータを覚えておき、いったんリターンキーを押しデータ入力画面に戻って、もう一度修正作業を行ってください。

修正データ書き込み処理

さて、データ修正ルーチンから帰り、これで修正処理は終りたいときには氏名入力に対して「999」をキーインすることになります。しかしこのとき前回は、メニュープログラム実行部分へジャンプしていましたが、このプログラムでは、メニューへ戻る前に修正データをファイルに書き込まなければなりません。

この部分の修正を行っておきましょう。

まず350行のジャンプ先を480から472に変更しておきます。

```
350 IF A$="999" THEN GOTO 472
```

そして改たに次のステートメントを追加入力します。

```
472 GOSUB *KAKIKOMI
474 '
```

ラベル名

データ書き込みのサブルーチンの**ラベル名**を

＊KAKIKOMI

とし、次を入力しましょう。

```
6000 '―――――――――――――――――――――――――――
6010  *KAKIKOMI
6020 '―――――――――――――――――――――――――――
6030 OPEN "2:telno1.dat" FOR OUTPUT AS #1
6040 FOR I=1 TO C
6050   WRITE #1,A$(I),B$(I)
6060 NEXT I
6070 CLOSE #1
6080 RETURN
```

これは**シーケンシャルファイル書き込みの「ひな型」**のようなものです。

書き込むデータ数は変数Cの中に格納されていますから、FOR〜NEXTループでこの数だけ繰り返します。

書き込みが終ったら、ファイルをクローズするのを忘れないようにしましょう。

以上で、データ修正プログラムは完成です。とりあえずディスクにセーブしておきましょう。

```
save "telno3.bas" ⏎
```

プログラムの実行

セーブしておけば安心です。一度プログラムを実行してみましょう。

例として「ヨシダ　ショウイン」を氏名、電話番号とも次のように変えてしまいましょう。

ヨシダ　ショウイン —— ヨシダ　シゲル
999-999-9999—— 123-123-1234

ではプログラムをRUNさせてください。氏名入力画面になりますから、「ヨシダ」と入力し ⏎ とします。すると確認のメッセージが現れますから、修正することにして [,] キーを押し、続いて氏名、電話番号を上のように変更してください。

次の画面は、新しいデータを入力しているところです。

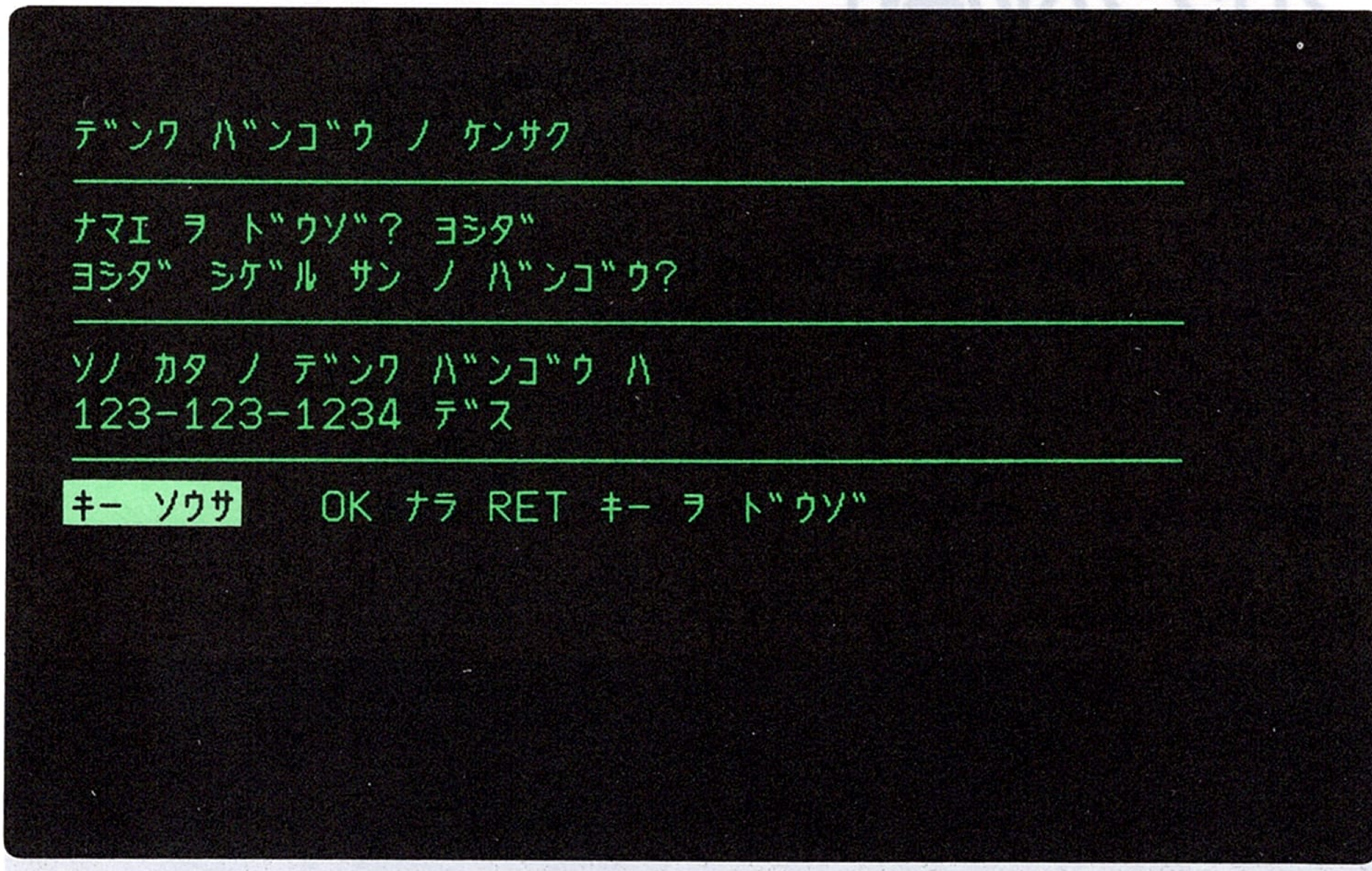
```
デ゛ンワ チョウ ノ シュウセイ
──────────────────────────────
ナマエ ヲ ド゛ウゾ゛? ヨシダ゛
ヨシダ゛ ショウインサン ノ シュウセイ?
──────────────────────────────
ナマエ        ヨシダ゛ ショウイン
シュウセイ: ? ヨシダ゛ シゲ゛ル
──────────────────────────────
デ゛ンワ      999-999-9999
シュウセイ: ? 123-123-1234■
──────────────────────────────
キー ソウサ   シュウセイ ナケレバ゛ RET キー ヲ ド゛ウゾ゛
```

この状態でリターンキーを押しますと、キーオペレーションメッセ

ージが変わります。そうしたらもう一度リターンキーを押してください。すると、次の修正データ入力となりますから、今度は「999」をキーインしてメニューに戻ります（このときに修正データは自動的にデータファイルに書き出されます）。

メニュー画面で4番の**データの検索を選択**してみましょう。

入力画面になったら、「ヨシダ」を入力してみましょう。最終画面は次のようになりました。



どうですか？　正確に修正されていましたね。

このあと、メニュー画面に戻って「終り」を選択しましょう。

次にプログラムリストをかかげますが、変更箇所がよくわかるように行番号の整理はしないでおきました。

```
100 '  file name ････> telno3.bas
110 '  1985.2.15
120 '  ﾃﾞﾝﾜ ﾁｮｳ ﾉ ｼｭｳｾｲ
130 '  save "telno3.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 PRINT "ﾃﾞﾝﾜ ﾁｮｳ ﾉ ｼｭｳｾｲ"
190 PRINT STRING$(40,149)
```

```
200 PRINT : PRINT
210 PRINT STRING$(40,149)
220 PRINT : PRINT
230 PRINT STRING$(40,149)
232 PRINT : PRINT
235 PRINT STRING$(40,149)
240 COLOR 4 : PRINT "ｷｰ ｿｳｻ"
250 COLOR 0
260 '
270 GOSUB *YOMU1
280 DIM A$(C),B$(C) : C=0
290 GOSUB *YOMU2
300 '
310 *LOOP
320 FOUND=0
330 LOCATE 8,11 : PRINT " ｵﾜﾘ ﾉ ﾄｷ ﾊ 999 "
340 LOCATE 0,2 : INPUT "ﾅﾏｴ ｦ ﾄﾞｳｿﾞ";A$
350 IF A$="999" THEN GOTO 472
360 GOSUB *KENSAKU
370 IF FOUND THEN GOSUB *SHUUSEI : GOTO 440
380 BEEP : BEEP
390 LOCATE 0,3 : PRINT SPC(50)
400 LOCATE 0,3 : COLOR 6
410 PRINT "ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !!"
420 FOR I=1 TO 2000 : NEXT I
430 COLOR 0
440 LOCATE 0,3 : PRINT SPC(50)
450 LOCATE 12,2 : PRINT SPC(50)
460 GOTO *LOOP
470 '
472 GOSUB *KAKIKOMI
474 '
480 CLS
490 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
500 FOR I=1 TO 5000 : NEXT I
510 RUN "menu02"
520 '
1000 '―――――――――――――――――――――――――――――――――
1010  *YOMU1
1020 '―――――――――――――――――――――――――――――――――
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
```

```
1050   INPUT #1,A$,B$
1060   C=C+1
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
2000 '-----------------------------------------
2010  *YOMU2
2020 '-----------------------------------------
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
2080 CLOSE #1
2090 RETURN
2100 '
3000 '-----------------------------------------
3010  *KENSAKU
3020 '-----------------------------------------
3030 FOR I=1 TO C
3040   IF INSTR(1,A$(I),A$) THEN GOSUB *KAKUNIN
3050   IF FOUND THEN T=I : RETURN
3060 NEXT I
3070 RETURN
3080 '
4000 '-----------------------------------------
4010  *KAKUNIN
4020 '-----------------------------------------
4030 LOCATE 8,11
4040 PRINT "  YES [,]   NO [.]"
4050 LOCATE 0,3 : PRINT A$(I);"ｻﾝ ﾉ ｼｭｳｾｲ?"
4060 I$=INKEY$ : IF I$="" THEN GOTO 4060
4070 IF I$="," THEN FOUND=-1 : GOTO 4100
4080 IF I$="." THEN GOTO 4100
4090 BEEP : BEEP : GOTO 4060
4100 LOCATE 8,11 : PRINT SPC(50)
4110 RETURN
4120 '
5000 '-----------------------------------------
5010  *SHUUSEI
5020 '-----------------------------------------
```

```
5030 LOCATE 8,11
5040 PRINT " ｼｭｳｾｲ ﾅｹﾚﾊﾞ RET ｷｰ ｦ ﾄﾞｳｿﾞ"
5050 LOCATE 0,5 : PRINT "ﾅﾏｴ          ";A$(T)
5060 PRINT "ｼｭｳｾｲ: "
5070 LOCATE 7,6 : INPUT A$
5080 IF A$="" THEN GOTO 5100
5090 A$(T)=A$
5100 LOCATE 0,8 : PRINT "ﾃﾞﾝﾜ          ";B$(T)
5110 PRINT "ｼｭｳｾｲ: "
5120 LOCATE 7,9 : INPUT B$
5130 IF B$="" THEN GOTO 5150
5140 B$(T)=B$
5150 LOCATE 8,11 : PRINT SPC(50)
5160 LOCATE 8,11
5170 PRINT " RET ｷｰ ﾃﾞ ﾂｷﾞ ﾉ ｼｭｳｾｲ ﾍ"
5180 I$=INKEY$ : IF I$="" THEN GOTO 5180
5190 LOCATE 8,11 : PRINT SPC(50)
5200 LOCATE 0,5 : PRINT SPC(50)
5210 LOCATE 0,6 : PRINT SPC(50)
5220 LOCATE 0,8 : PRINT SPC(50)
5230 LOCATE 0,9 : PRINT SPC(50)
5240 RETURN
5250 '
6000 '--------------------------------------------------
6010 *KAKIKOMI
6020 '--------------------------------------------------
6030 OPEN "2:telno1.dat" FOR OUTPUT AS #1
6040 FOR I=1 TO C
6050   WRITE #1,A$(I),B$(I)
6060 NEXT I
6070 CLOSE #1
6080 RETURN
```

3・7　データを印刷する

初期画面から印刷までのあらすじ

私たちはこれまで電話番号データファイルの作成をし、データの追加をしたり修正をしたり、また希望する人の電話番号を知ることにも成功しました。

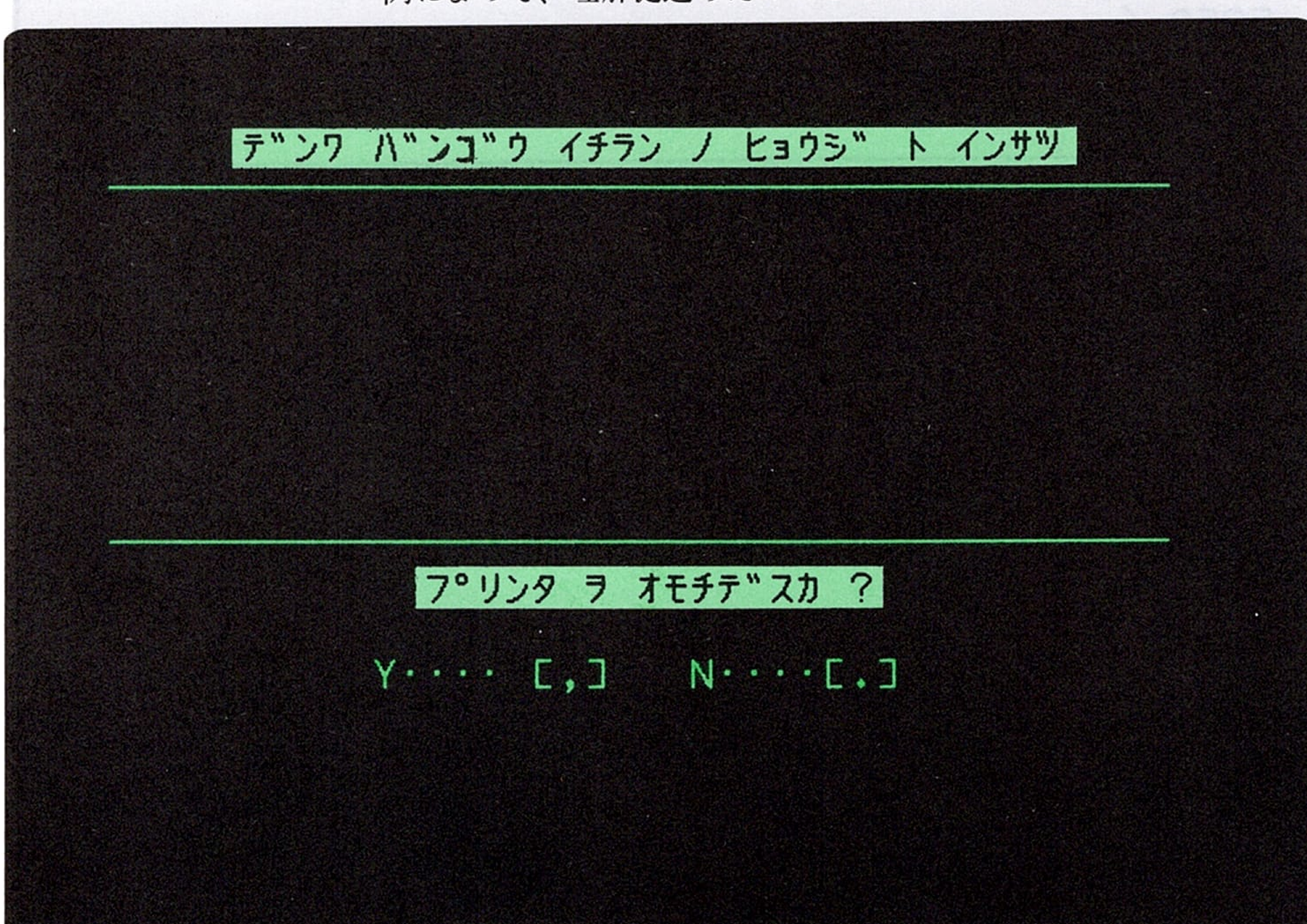
しかし、これらはすべてディスプレイ上で行ってきましたし、登録データすべてをひと目で見るというわけにはまいりませんでした。

また、データファイル作成中は、プログラムに入力ミスがないかどうかのルーチンもつけてはいますが、キー操作は人間のやることです、けっこう間違いがあるものです。

この節では、**登録データをプリンタで印刷**し、もう一度この目で確かめてみましょう。

また、いままではディスクのデータを読み出して画面に表示はできましたが、それでもまだ、ディスクにほんとうに書き込まれているという実感がなかなかもてませんでした。しかし、プリンタで実際に印刷されてしまえば、信じざるをえないというものです。

例によって、理解促進のためにこのプログラムのあらすじを説明し



ておきましょう。

まず、このプログラムをRUNすると前ページのような初期画面になり、プリンタを使用するかどうか聞いてきます。

なぜ**プリンタ有無の確認**を入れたかと申しますと、読者の中にはプリンタをお持ちでない方もいると思います。そんな方には、画面へデータを出力して、その理屈を理解していただきたいからです。

また、プリンタを持っている方は

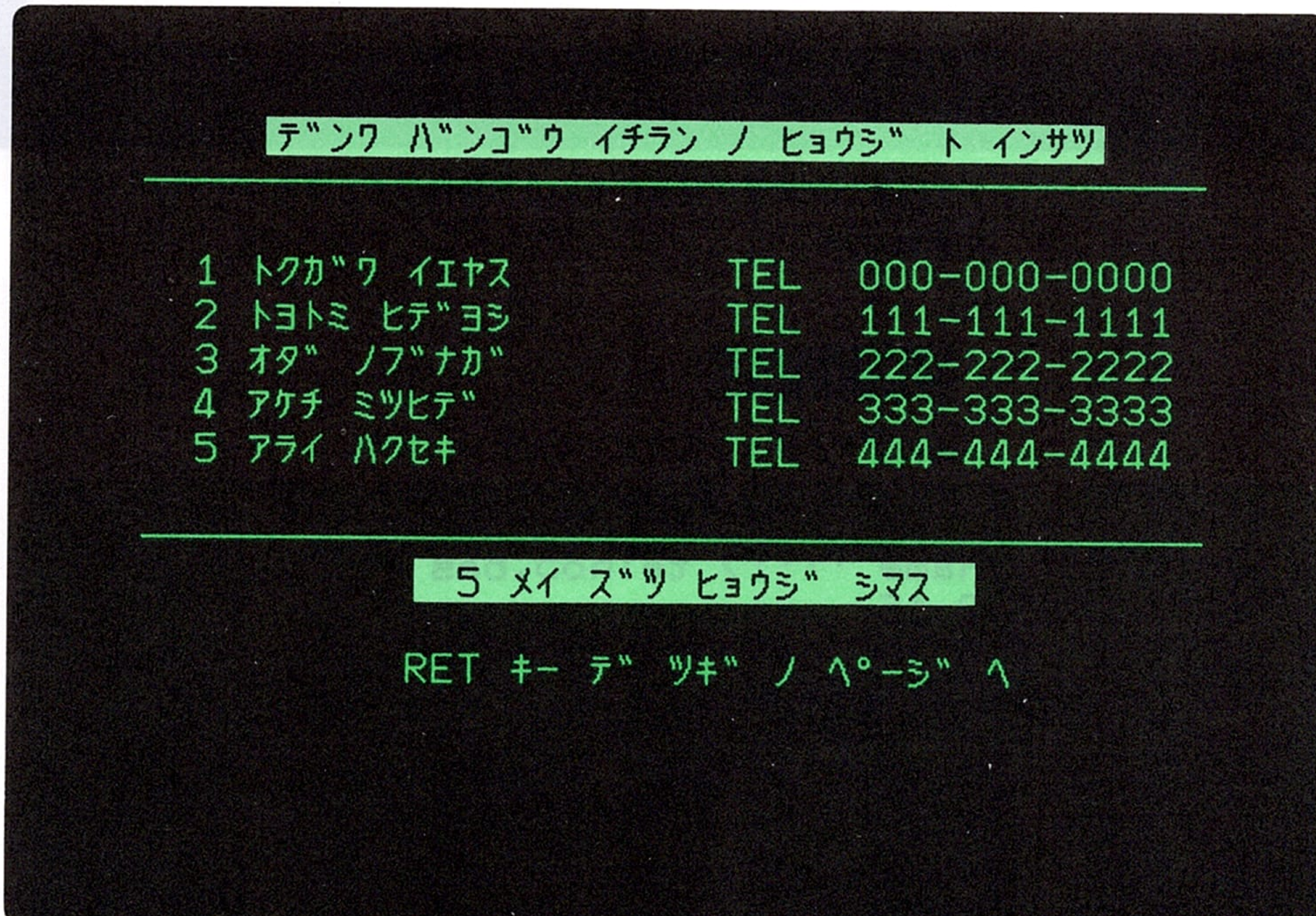
画面出力

と

プリンタ出力

の両方を行うことができます。

前ページの画面でいずれを選択しても、まず次のように**5人分の電話番号データがディスプレイに出力**されます。



プリンタを使用している方は、この画面になるとすぐに5人分のデータがプリンタへ印字されます。そしてリターンキーを押すと次の5人分のデータが表示されます。

私たちは、これまで10人分のデータを入力してきましたから、ディスプレイは2ページの切り換えで終り、いつものとおりメニュープロ

グラムが実行されます。

次に示すのはプリンタへの出力例です。

```
 1  トクガ゛ワ イエヤス         TEL  000-000-0000
 2  トヨトミ ヒデ゛ヨシ         TEL  111-111-1111
 3  オダ゛ ノブ゛ナガ゛         TEL  222-222-2222
 4  アケチ ミツヒデ゛           TEL  333-333-3333
 5  アライ ハクセキ             TEL  444-444-4444
 6  カイバ゛ラ エッケン         TEL  555-555-5555
 7  カイホ セイリョウ           TEL  666-666-6666
 8  ヨシダ゛ ケンコウ           TEL  777-777-7777
 9  サトウ イッサイ             TEL  888-888-8888
10  ヨシダ゛ シゲ゛ル           TEL  123-123-1234
```

このような結果が得られることを期待して、さあコーディングに移りましょう。

初期設定

次のプログラムを入力してください。

```
100 ' file name ····> telno5.bas
110 ' 1985.2.15
120 ' デ゛ンワ バ゛ンゴ゛ウ イチラン ノ ヒョウジ゛ ト インサツ
130 ' save "telno5.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 ED1$="### &←――スペース15文字分――→&"
180 ED2$=" TEL  &←――スペース14文字分――→&"
```

いつもと同じプログラムヘッドですが、170行、180行がいつもと違います。

おわかりの方も多いと思いますが、一応説明しておきましょう。

これは、**数値とか文字を指定した書式で出力したいときに使う編集**

用の文字列です。**PRINT(プリント)文**、**LPRINT(エルプリント)文**とともに用い、いろいろな書式が指定できます。

PRINT文
LPRINT文

ここでは二、三の例をあげるにとどめ、そのほかはみなさんが必要に応じてリファレンスマニュアルを参照していただきたいと思います。

たとえば、

```
A$="PC-9801M" : B$="PC-8801MK2"
```

というときに

```
PRINT USING "&      &" ; A$ ↵
PRINT USING "&      &" ; B$ ↵
```

スペースを意味します

PRINT USING文

〈注〉プリンタへ出力したいときは「L」をつけて LPRINT USING とします。

を実行すると、それぞれ、

```
PC-9801M
PC-8801M
```

のように出力されます。つまり、**&と&ではさまれたスペースの数プラス2**、上の例でいくと8文字分の中に左寄せで制御出力されますから、B$の内容は右端の2文字が無視されてしまいます。

ですから、同じパターンの出力制御をしたいときは少し余裕をもって、

```
PRINT USING "&          &" ; A$
```

などとしておくとよいでしょう。この場合には12文字の幅で左づめに出力され、余ったところにはスペースが埋められます。

そして、この**書式指定用の文字列**

書式指定用の文字列

```
"&          &"
```

は、文字型変数に代入しておくことができますから、あらかじめ

```
ED$="&          &
```

などとしておくと、プログラム中で

```
PRINT USING ED$ ; A$
```

といった使い方ができるので大変便利です。

なお、**シャープ(#)**は数値の書式指定をするときに、たとえば

```
PRINT USING "####" ; 35 ↵
```

などとすると、

␣␣35

というように、右づめで出力されます。

説明はこのくらいにして、あとはプログラムの中で実際に使ってみることにしましょう。

初期画面部分の入力

130ページに示した初期画面を出力するステートメントです。

```
190 '
200 LOCATE 24,0 : COLOR 4
210 PRINT " ﾃﾞﾝｸﾜ ﾊﾞﾝｺﾞｳ ｲﾁﾗﾝ ﾉ ﾋｮｳｼﾞ ﾄ ｲﾝｻﾂ"
220 COLOR 0 : LOCATE 20,1
230 PRINT STRING$(40,149)
240 LOCATE 20,9
250 PRINT STRING$(40,149)
260 LOCATE 30,10 : COLOR 6
270 PRINT "  ﾌﾟﾘﾝﾀ ｦ ｵﾓﾁﾃﾞｽｶ ? "
280 COLOR 0 : LOCATE 30,12
290 PRINT "Y････ [,]   N････[.]"
300 I$=INKEY$ : IF I$="" THEN GOTO 300
310 IF I$="," THEN INSATSU=-1 : GOTO 330
320 IF I$<>"." THEN BEEP : BEEP : GOTO 300
330 LOCATE 20,10 : PRINT SPC(50)
340 LOCATE 20,12 : PRINT SPC(50)
350 '
```

210行の見出しの表示は、前回と同じように**COLOR文**を使って**反転表示**するようにしました。画面構成にアクセントをつけるのに大変役立ちます。

横に引いた罫線は**STRING$関数**で処理しました。

プリンタ有無の確認メッセージも反転表示出力をさせています。

メッセージに対しての回答はテンキー側の [,] か [・] キーを使いますが、なぜ [Y] および [N] を使わなかったかといいますと、ほかの部分で [↵] キーを使う場合がありますので、オペレーションのためのキーを一箇所にまとめておくと使いがってがよいからです。

次に、310行の

INSATSU=−1

について説明をしておきます。

次のようにダイレクトモードでキーインしてみてください。

```
A=1
Ok
? A=1
-1
Ok
B=2
Ok
? B=1
 0
Ok
■
```

最初に、

A=1 ⏎

としましたから、変数Aには1が代入されました。ということは次の

A=1

という式は、すでにAに1が代入されていますから左辺は1、また右辺は当然1ですから、

1=1

となり、この式は「**真**」となります。「1が1に等しい」とは当たり前のことですね。そして「真」である証拠として 真

? A=1 ⏎

とすると、**−1**という値を返してきます。次に、

B=2

としますと変数Bには2が代入されます。ですから、

B=1

という式は実際には

2=1

となり、この等式は当然成立しませんから「**偽**」となります。そして 偽

「偽」のときには、その値として**0**を返します。

ところで話を元に戻しますが、

INSATSU=−1

というのは、**「もしプリンタを持っている場合にはINSATSUという変数の内容を−1にする、つまり真にする」**ことを意味します。反対にプリンタを持っていなければINSATSUの内容は初期値のままですから0です。そしてこのことをあとで使いますので頭のすみに入れておいてください。

データの読み出し

この部分は前節とまったく同じです。

```
360 GOSUB *YOMU1
370 DIM A$(C),B$(C) : C=0
380 GOSUB *YOMU2
390 '
```

```
1000 '―――――――――――――――――――――――――
1010  *YOMU1
1020 '―――――――――――――――――――――――――
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
1050   INPUT #1,A$,B$
1060   C=C+1
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
2000 '―――――――――――――――――――――――――
2010  *YOMU2
2020 '―――――――――――――――――――――――――
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
2080 CLOSE #1
2090 RETURN
2100 '
```

データファイルをオープンしてデータのカラ読みをし、配列宣言をしてからもう一度データを読み出し、今度は配列にしまいます。

この部分はすでに説明ずみですので省略します。

画面表示とプリンタへの出力

さて次はデータを画面またはプリンタへ出力しますが、その前にキー操作のメッセージを変えておきます。そして、こうしておいてから出力のサブルーチンへジャンプさせます。

```
400 LOCATE 26,10 : COLOR 4
410 PRINT SPC(6);"5 ﾒｲ ｽﾞﾂ ﾋｮｳｼﾞ ｼﾏｽ ";SPC(5)
420 COLOR 0 : LOCATE 30,12
430 PRINT "RET ｷｰ ﾃﾞ ﾂｷﾞﾉ ﾍﾟｰｼﾞ ﾍ "
440 '
450 GOSUB *SHUTSURYOKU
460 '
```

```
3000 '————————————————————————————
3010  *SHUTSURYOKU
3020 '————————————————————————————
3030 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
3040 FOR I=1 TO C STEP 5
3050   FOR J=1 TO 5
3060     M=I+J-1
3070     LOCATE 20,2+J
3080     PRINT USING ED1$;M,A$(M);
3090     IF NOT INSATSU THEN GOTO 3110
3100     LPRINT USING ED1$;M,A$(M);
3110     PRINT USING ED2$;B$(M)
3120     IF NOT INSATSU THEN GOTO 3140
3130     LPRINT USING ED2$;B$(M)
3140   NEXT J
3150   I$=INKEY$ : IF I$="" THEN GOTO 3150
3160   FOR K=3 TO 7
3170     LOCATE 20,K : PRINT SPC(50)
3180   NEXT K
3190 NEXT I
3200 RETURN
3210 '
```

STEP(ステップ)

NOT(ノット)

PRINT USING

まず3030行で**ブザー音**をならしてから、このサブルーチンに入ります。

変数ＪのFOR〜NEXTループで５名の氏名と電話番号を表示し、変数ＩのFOR〜NEXTループで次のパターンへ移ります。

５名ずつの表示ですから変数Ｉのきざみは５、**STEP５**をつければよいですね。この場合Ｉは、1、6、11、……というように５きざみで変化していきます。そしてこの間の落ちている数、2、3、4、5、7、8、9、10、……などは、ループ変数Ｊの値を加えて調整し、1、2、3、4、5、6、7、……、10、11、……という連続した整数列を作り出しています。この値は3060行で計算され、Ｍに代入されます。このＭの値を配列変数Ａ＄、Ｂ＄の添字に使うわけですね。

ところで、

IF NOT INSATSU THEN〜

というステートメントがありますが、前に説明した論理値を思い出してください。**NOT**は否定を表す論理演算子です。つまり上のステートメントの意味は、

もし、変数INSATSUの内容が真でないならば〜

あるいは、

もし、変数INSATSUの内容が−１でないならば〜

ということになります。そしてこの条件が満たされたときにTHEN以下を実行します。

変数INSATSUの内容が−１でないとき、つまり０のとき（プリンタを持っていないとき）THEN文以下が実行されプリンタへの出力行3100行、3130行を飛びこえてしまいます。

3150行はキー入力待ちで、何かキーが押されるまでは、この行でとどまっています。

つまり、どんなキーでもよいですから、キー入力があるとこの行を抜けて、3160〜3180行のループで表示部分をクリアしてから次の５名が表示されます。

画面の表示には、前に説明した**PRINT USING文**を使っています。

さて、これでデータ表示および印刷のサブルーチンは完成です。このサブルーチンから戻ったら次はメニューへ戻るステートメントを入力しておきましょう。次ページにプログラムを示します。

```
470 CLS
480 LOCATE 30,3
490 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
500 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
510 FOR I=1 TO 2000 : NEXT I
520 RUN "menu02"
530 '
```

そして、とりあえずプログラムをセーブしておきましょう。

```
save "telno5.bas" ⏎
```

プログラムの実行

それでは、さっそく実行してみたいのですが、その前にもう少しデータを入れておきましょう。

メニュープログラムをRUNしてください。

```
run "menu02" ⏎
```

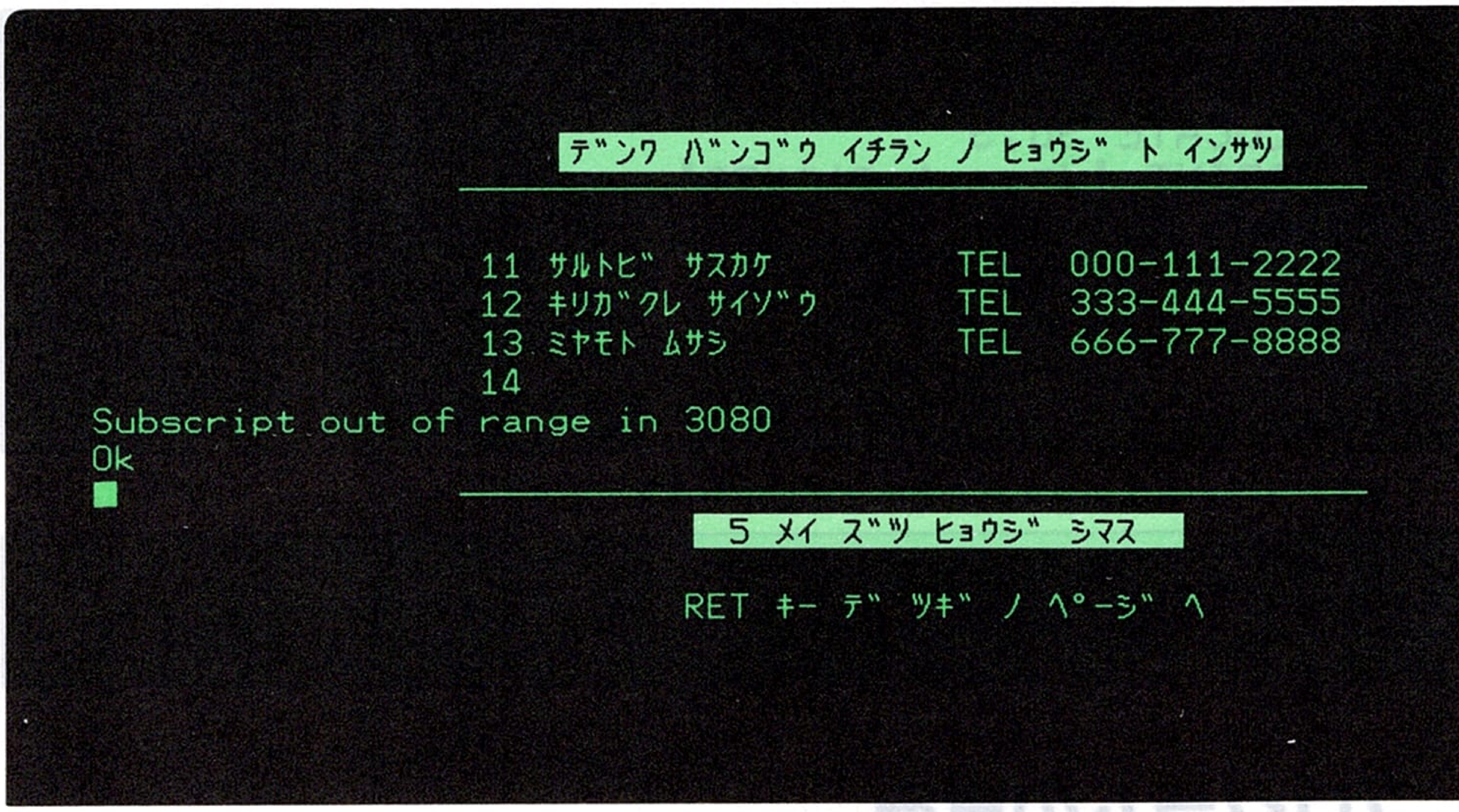
メニュー画面が現れましたら2番の**「データの追加」**を選択し、次のデータを入力してください。

サルトビ　サスケ…………000－111－2222
キリガクレ　サイゾウ……333－444－5555
ミヤモト　ムサシ…………666－777－8888

入力が終りましたら再びメニュー画面に戻り、5番の**「データの印刷」**を選択します。すると、この節で作ったプログラムが実行され、プリンタ確認のメッセージにはどちらでもいいですから答えてください。131ページのような画面が表示されましたね。

リターンキーを押すと続いて2ページ目が、さらにもう一度リターンキーを押すと3ページ目が出力され、たったいま追加入力した3名のデータが表示されますが、ここでブザー音が鳴ってエラーメッセージが表示されました。

画面は次ページのようになってストップしてしまいました。



このエラーメッセージの意味はわかりますね。そうです、

配列の添字が制限範囲を超えていますよ

ということです。

136ページのリストの370行で、配列A＄、B＄の大きさは登録数だけ確保されていることがわかります。つまり、私たちは13番目の人まで登録したのですから配列の添字は最大13でなければなりません。ところが上の画面を見ますと14という数が表示されています。

137ページのリストの3080行をごらんください。ここで、

A$（14）

を表示しようとしてエラーとなったのです。

これを防ぐには3065行として次のステートメントをキーインします。

```
3065      IF M>C THEN GOTO 3150
```

Mの内容をチェックし、それが登録数を超えたら強制的にループから抜け出してしまいます。

さあ、今度は正常に動作すると思いますが、ステートメントの追加をしたので、プログラムをもう一度セーブし直しておきましょう。

```
save "telno5.bas"
```

プログラム総リストを次にかかげます。

```
100 ' file name ････> telno5.bas
110 ' 1985.2.15
120 ' ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ｲﾁﾗﾝ ﾉ ﾋｮｳｼﾞ ﾄ ｲﾝｻﾂ
130 ' save "telno5.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 ED1$="### &               &"
180 ED2$=" TEL  &               &"
190 '
200 LOCATE 24,0 : COLOR 4
210 PRINT " ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ｲﾁﾗﾝ ﾉ ﾋｮｳｼﾞ ﾄ ｲﾝｻﾂ"
220 COLOR 0 : LOCATE 20,1
230 PRINT STRING$(40,149)
240 LOCATE 20,9
250 PRINT STRING$(40,149)
260 LOCATE 30,10 : COLOR 6
270 PRINT "  ﾌﾟﾘﾝﾀ ｦ ｵﾓﾁﾃﾞｽｶ ? "
280 COLOR 0 : LOCATE 30,12
290 PRINT "Y････ [,]   N････[.]"
300 I$=INKEY$ : IF I$="" THEN GOTO 300
310 IF I$="," THEN INSATSU=-1 : GOTO 330
320 IF I$<>"." THEN BEEP : BEEP : GOTO 300
330 LOCATE 20,10 : PRINT SPC(50)
340 LOCATE 20,12 : PRINT SPC(50)
350 '
360 GOSUB *YOMU1
370 DIM A$(C),B$(C) : C=0
380 GOSUB *YOMU2
390 '
400 LOCATE 26,10 : COLOR 4
410 PRINT SPC(6);"5 ﾒｲ ｽﾞﾂ ﾋｮｳｼﾞ ｼﾏｽ ";SPC(5)
420 COLOR 0 : LOCATE 30,12
430 PRINT "RET ｷｰ ﾃﾞ ﾂｷﾞ ﾉ ﾍﾟｰｼﾞ ﾍ "
440 '
450 GOSUB *SHUTSURYOKU
460 '
470 CLS
480 LOCATE 30,3
490 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
500 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
510 FOR I=1 TO 2000 : NEXT I
520 RUN "menu02"
530 '
```

```
1000 '
1010  *YOMU1
1020 '
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
1050   INPUT #1,A$,B$
1060   C=C+1
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
2000 '
2010  *YOMU2
2020 '
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
2080 CLOSE #1
2090 RETURN
2100 '
3000 '
3010  *SHUTSURYOKU
3020 '
3030 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
3040 FOR I=1 TO C STEP 5
3050   FOR J=1 TO 5
3060     M=I+J-1
3065     IF M>C THEN GOTO 3150
3070     LOCATE 20,2+J
3080     PRINT USING ED1$;M,A$(M);
3090     IF NOT INSATSU THEN GOTO 3110
3100     LPRINT USING ED1$;M,A$(M);
3110     PRINT USING ED2$;B$(M)
3120     IF NOT INSATSU THEN GOTO 3140
3130     LPRINT USING ED2$;B$(M)
3140   NEXT J
3150   I$=INKEY$ : IF I$="" THEN GOTO 3150
3160   FOR K=3 TO 7
3170     LOCATE 20,K : PRINT SPC(50)
3180   NEXT K
3190 NEXT I
3200 RETURN
3210 '
```

3・8 データの並べかえ

前節で電話番号のリストを無事印刷することができましたが、これは私たちがキーボードから入力した登録順になっていて、氏名が50音順になっていません。これを50音順に並べかえて印刷すれば、何かと使い勝手もよくなるのではないかと思います。

このように**順序を並べかえること**を**ソート**と呼び、ビジネスの分野では、数値を大きい順に並べたり、小さい順に並べたり、とにかく、よく使われます。

並べかえ

ソート

私たちも、ここでソートにチャレンジしてみましょう。

下に示したのは並べかえを行った出力結果です。132ページの出力結果と比較してみてください。50音順にきれいに整理されて打ち出されています。

並べかえはプログラムの組み方によってスピードに大きな差が出てきます。本節での並べかえ（ソート）はデータ数が多くなるとだいぶ時間がかかりますが、理屈を学びとってください。

```
 1 アケチ ミツヒデ゛        TEL  333-333-3333
 2 アライ ハクセキ          TEL  444-444-4444
 3 オダ゛ ノブ゛ナガ゛      TEL  222-222-2222
 4 カイハ゛ラ エッケン      TEL  555-555-5555
 5 カイホ セイリョウ        TEL  666-666-6666
 6 キリガ゛クレ サイゾ゛ウ  TEL  333-444-5555
 7 サトウ イッサイ          TEL  888-888-8888
 8 サルトビ゛ サスケ        TEL  000-111-2222
 9 トクガ゛ワ イエヤス      TEL  000-000-0000
10 トヨトミ ヒデ゛ヨシ      TEL  111-111-1111
11 ミヤモト ムサシ          TEL  666-777-8888
12 ヨシダ゛ ケンコウ        TEL  777-777-7777
13 ヨシダ゛ シゲ゛ル        TEL  123-123-1234
```

〈注〉下のプログラムは説明のためですからキーインしなくてもかまいません。キーインする場合はNEWコマンドを実行してから入力してください。

この電話帳リストでのソートは、数値ではなく文字を並べかえることになりますが、原理は同じですので、また、そのほうが理解しやすいと思いますので、数値を例にとって説明します。

次に簡単な**ソートプログラム**をかかげます。

```
100 ' sorting
110 '
120 FOR I=1 TO 5
130   READ A(I)
140   PRINT A(I);"  ";
150 NEXT I
160 PRINT
170 '
180 FOR I=1 TO 4
190 K=I
200   FOR J=I+1 TO 5
210     IF A(J)<A(K) THEN K=J
220   NEXT J
230   SWAP A(I),A(K)
240 NEXT I
250 FOR I=1 TO 5
260   PRINT A(I);"  ";
270 NEXT I
280 DATA 4,1,5,3,2
```

- 130 ← ソートするデータを配列に読み込む
- 140 ← ソート前のデータを画面に表示
- 160 ← これを入れておかないとソート後のデータが続いて表示されてしまう
- 210 ← 小さい値があればポインタKをその値のところに合わせる
- 230 ← ポインタKの指し示す配列A(K)の中身と配列A(I)の中身を入れかえる
- 260 ← ソート後のデータを画面に表示
- 280 ← ソートするデータ

バブルソート

このソートはバブルソートと呼ばれるもので、もっとも基本的なものです。

方法は、まずソートするデータを配列に読み込み、ある1つのデータを基準にして、次々と他のデータと大きさを比較していきます。

たとえば、小さい順に並べるのでしたら、基準となっている数より小さい数が現れたら、今度はその数を基準にし、次のデータと比較していきます。

このような操作を次々とくりかえし、**最後に基準になっているデータを先頭のほうへもってきます。**

ポインタ

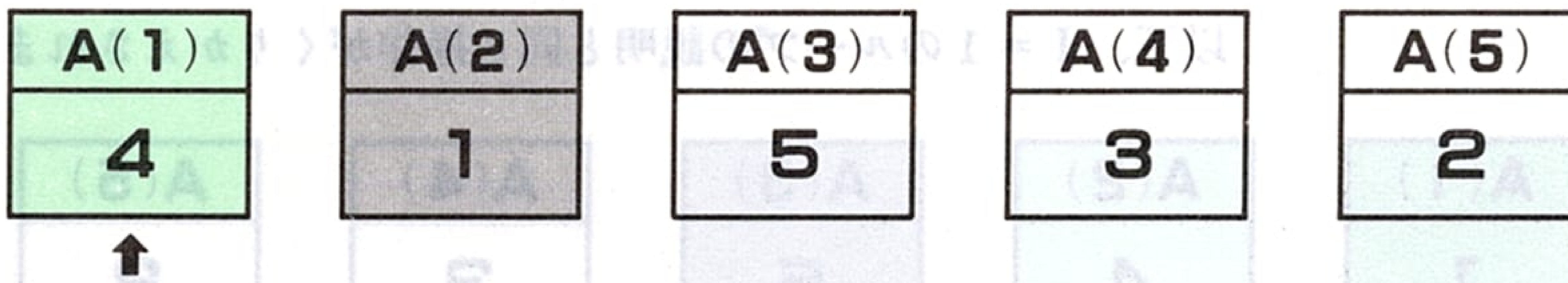
この比較する基準となるデータは、ポインタによって指し示されています。上の例でいいますと、Kがポインタになっています。

SWAP

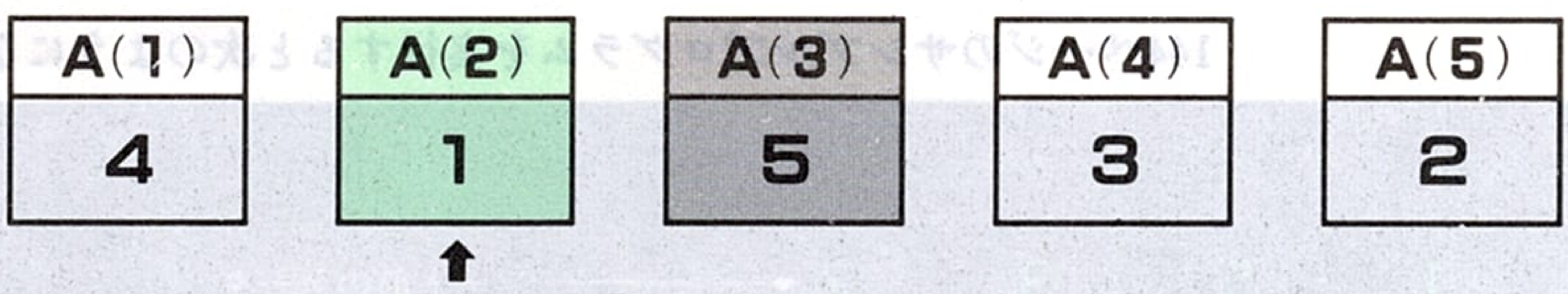
小さいものを先頭にもってくる作業は**SWAP(スワップ)文**で行います。

230行のSWAPの意味は、A(I)とA(K)の内容を入れかえるということです。上のプログラムにそって、もう少し説明を加えます。

最初のループ、つまりI＝1でJのループに入ったときの状態は次のようになっています。K＝1ですから**ポインタは矢印の所**にあります。



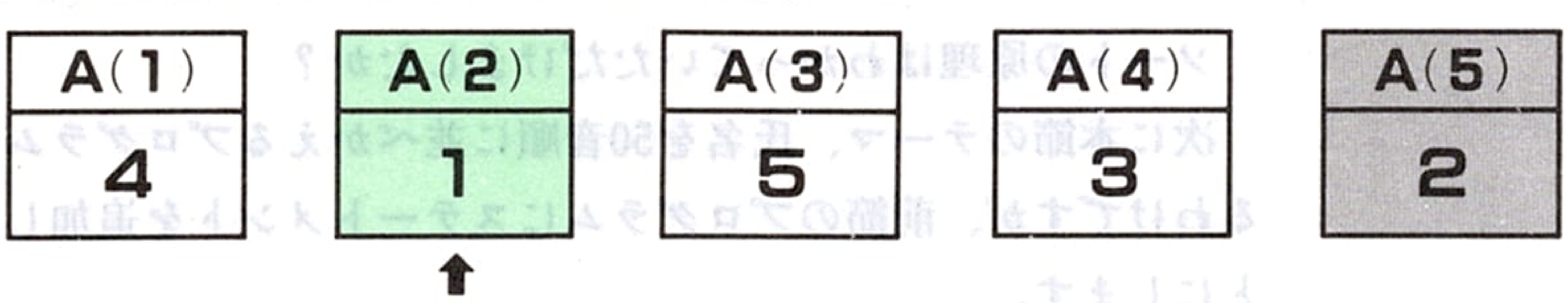
ここでA(1)とA(2)の内容を比較します。4＞1、つまりポインタが指し示すA(1)の内容より小さい値がでてきたわけですから、ポインタをその小さいほうの値のところに合わせます。この場合2番目ですからK＝2になります（210行のK＝Jの操作をしたわけです）。**ポインタが移動**して次のような状態になりました。



次に、ポインタが指し示すA(2)の内容とA(3)の内容を比較します。**この場合は1＜5ですからポインタは動きません。**

次に、やはりポインタが指し示すA(2)の内容とA(4)の内容を比較します。この場合も1＜3ですから、ポインタは相変わらずA(2)のところを指し示しています。

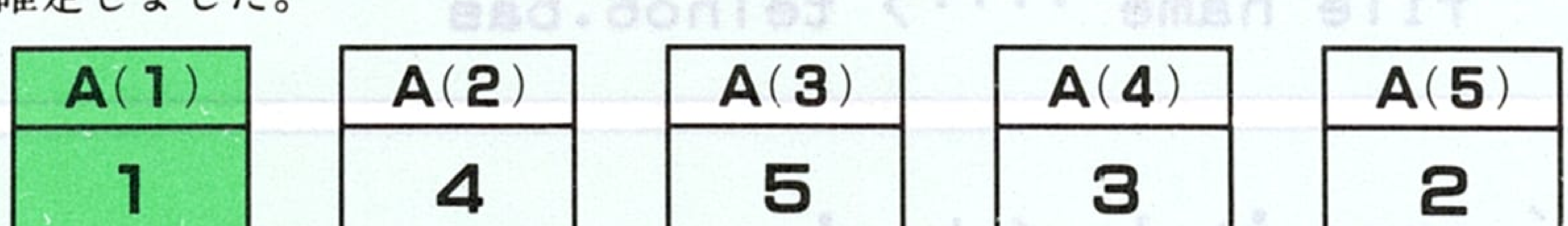
最後、現在のポインタ位置A(2)の内容とA(5)の内容を比較します。このときの状態は次のようになっています。



当然1＜2ですから、ポインタの位置は変わりません。

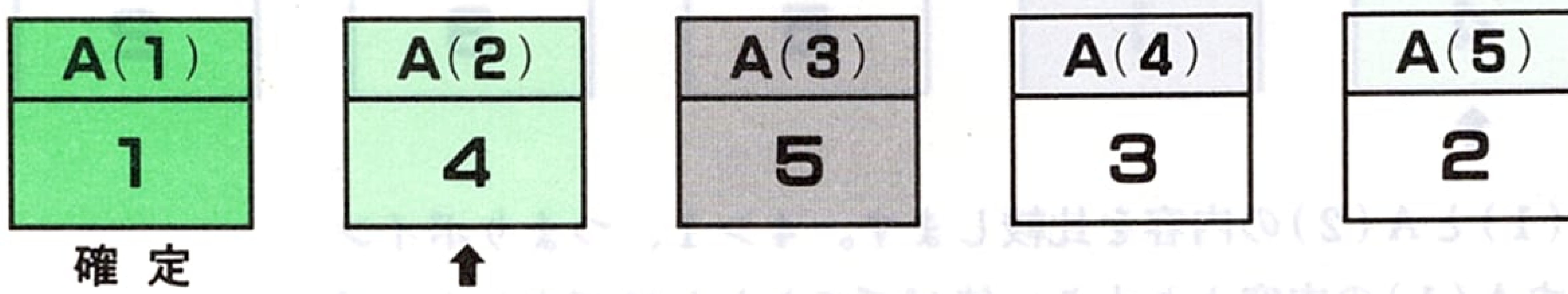
これでI＝1のときのJのループを1回終えたわけです。ですから次のIのループに行く前に、最小値、つまりポインタが指し示している配列の内容をA(1)の内容と入れかえなければなりません。この操作を230行の**SWAP文**でやっています。

入れかえた状態は次のようになり、A(1)の内容は5つの数の中の最小値と確定しました。



次にI＝2のループでは、最初のポインタKはK＝Iですから2にセットされ、図のような状態になり、ポインタの指し示すA(2)の内容とA(3)の内容を比較することから始まります。

以下、I＝1のループの説明と同じ操作がくりかえされます。



このループでは、ポインタは最終的にはA(5)を指し示すことになり、A(2)の内容とA(5)の内容がSWAP文により入れかわります。

144ページのサンプルプログラムを実行すると次のようになります。

```
run
 4     1     5     3     2   ←―――― ソート前のデータ
 1     2     3     4     5   ←―――― ソート後のデータ
Ok
■
```

メニューを作る

ソートの原理はわかっていただけましたか？

次に本節のテーマ、氏名を50音順に並べかえるプログラム作りに入るわけですが、前節のプログラムにステートメントを追加して作ることにします。

まず、前節のプログラムをロードしてください。

```
load "telno5.bas"↵
```

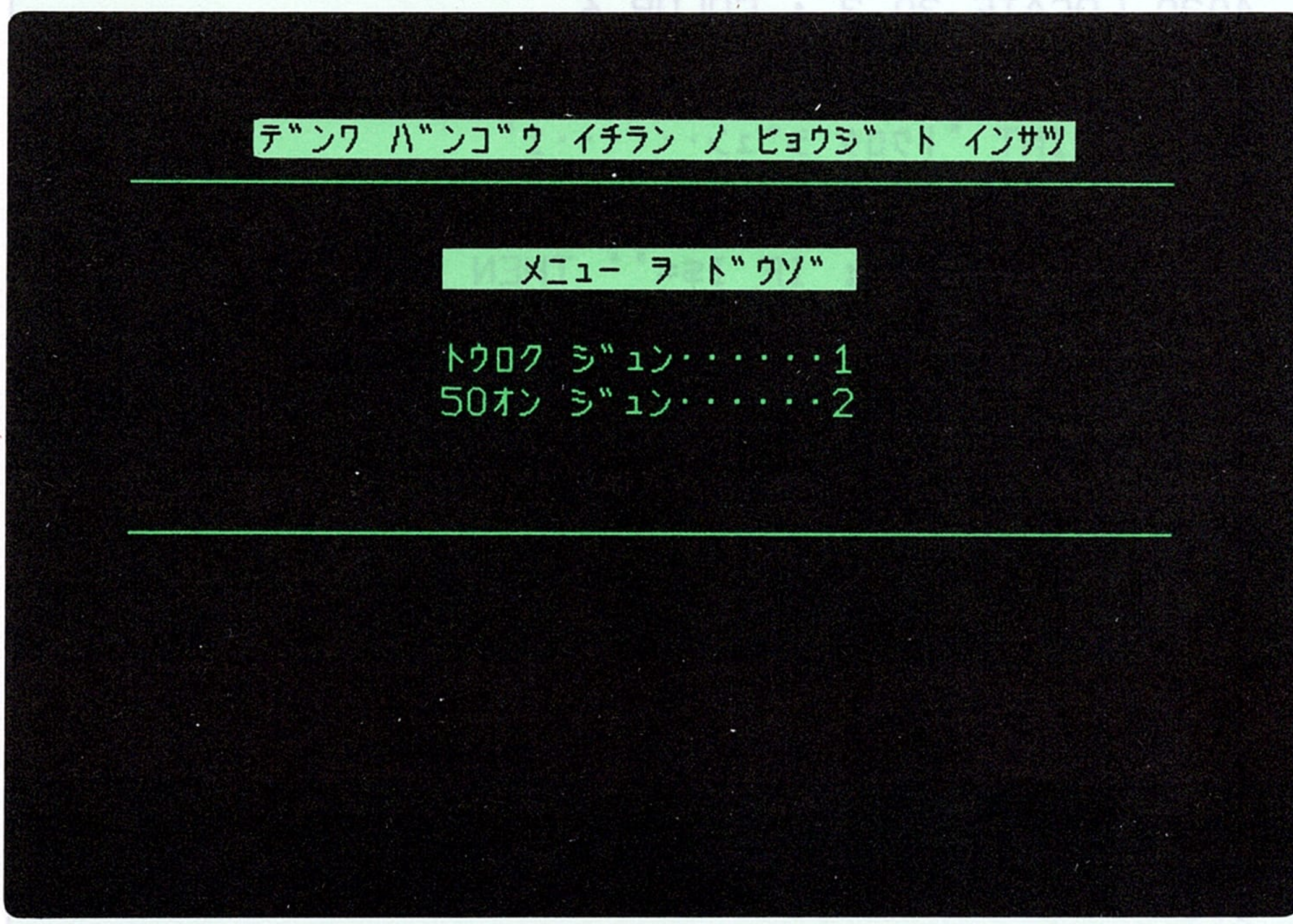
ロードできたら、行番号100と130のファイル名を変更しておきます。

```
100 ' file name ····> telno6.bas
```

```
130 ' save "telno6.bas"
```

これで準備ができました。

ところで、このプログラムに並べかえだけでなく登録順の出力もできるように2つのオプションを持たせたいと思います。そのために最初、次のような**メニュー画面**を表示するようにしましょう。



この処理は、プリンタ使用の確認の前に入れ、サブルーチンへジャンプさせて行うことにします。

次のプログラムを入力してください。

```
255 GOSUB *MENU
```

次に、このサブルーチンの入力ですが、4000行からとしてプログラムリストを次ページに示します。

とくに問題のところはないと思いますが、メニュー番号の入力は、INKEY＄で受けていますので4000行、4110行で行っているようにダブルコーテーションでくくるのを忘れないでください。数字ではありません。ストリングです。

それからもう1つ、 2 キーを押し「50音順」が選択されると136ページで説明したようにNARABEKAEという変数を使って、その内

```
4000 '―――――――――――――――――――――――――――――――
4010  *MENU
4020 '―――――――――――――――――――――――――――――――
4030 LOCATE 30,3 : COLOR 6
4040 PRINT "     ﾒﾆｭｰ ｦ ﾄﾞｳｿﾞ      "
4050 LOCATE 32,5 : COLOR 0
4060 PRINT "ﾄｳﾛｸ ｼﾞｭﾝ･････1"
4070 LOCATE 32,6
4080 PRINT "50ｵﾝ ｼﾞｭﾝ･････2"
4090 I$=INKEY$ : IF I$="" THEN GOTO 4090
4100 IF I$="2" THEN NARABEKAE=-1 : GOTO 4120
4110 IF I$<>"1" THEN BEEP : GOTO 4090
4120 LOCATE 25,3 : PRINT SPC(50)
4130 LOCATE 25,5 : PRINT SPC(50)
4140 LOCATE 25,6 : PRINT SPC(50)
4150 RETURN
4160 '
```

容を－1、つまり「**真**」にしています。そしてこれを、並べかえをするかしないかの条件判断に使います。次を入力してください。

```
395 IF NARABEKAE THEN GOSUB *SORT
396 '
```

ソートサブルーチン

もし**NARABEKAEが真なら**、並べかえのサブルーチンへジャンプします。次の**サブルーチン**を入力しましょう。

```
5000 '―――――――――――――――――――――――――――――――
5010  *SORT
5020 '―――――――――――――――――――――――――――――――
5030 FOR I=1 TO C-1
5040 K=I
5050   FOR J=I+1 TO C
5060     IF A$(J)<A$(K) THEN K=J
5070   NEXT J
5080   SWAP A$(I),A$(K)
5090   SWAP B$(I),B$(K)
5100 NEXT I
5110 RETURN
```

ソートサブルーチンのしくみについては先ほど説明したので省略しますが、先の場合は数字でしたから大小がはっきりわかりましたが、文字の場合は大小比較をどのように行っているのかを述べておきます。

キャラクタコード

結論から申しますと、キャラクタにはそれぞれに対応した**キャラクタコード**があり、その大小によって比較します。

79ページのキャラクタコード表をごらんください。「ア」が&HB1、「ン」が&HDD、10進数になおして177から221までとなっています。この数字の大小で、どちらの文字が先にあるかわかるわけです。つまり、文字のソート数字のソートに帰するわけです。

ASC関数

みなさんは**ASC（アスキー）関数**をごぞんじですね。キャラクタをパラメータとして与えると、それに対応した10進数のキャラクタコードに変換してくれます。

次のようなテストを行ってみてください。

```
? asc("ア"),asc("カ"),asc("ラ")
 177             182              215
Ok
```

キャラクタコード

この結果によっても、文字に大小があることがおわかりいただけたと思います。

以上で、本節でのステートメントの追加は終りです。実行してテストする前に、新しいプログラムとしてセーブしておきましょう。

```
save "telno6.bas" ⏎
```

セーブできたら実行してみてください。メッセージに従ってキー入力すればよいのですから簡単です。プリンタをお持ちの方は、143ページのような、並べかえされたデータが出力されるはずです。

ところで、89ページのメニューで５番の**「データ　ノ　インサツ」**を選択すると、前節の

telno5.bas

が実行されますが、これを、本節で作った

telno6.bas

が実行されるように直しておきましょう。

メニュープログラムをロードし、690行のDATA文のプログラム名を

を変更しておきましょう。

```
load "menu02." ⏎
```

```
690 DATA デ゛ータ ノ インサツ ・・・・・・・・・・・・・・5,telno6.bas"
```

修正が終ったら**メニュープログラムをセーブ**しておきます。

```
save "menu02" ⏎
```

メニュープログラムを実行して、第3章で作ったプログラムが正常に動作するかどうかテストしてみてください。

最後に本節でのプログラムリストをかかげておきます。

```
100 ' file name ・・・・> telno6.bas
110 ' 1985.2.15
120 ' デ゛ンワ バ゛ンゴ゛ウ イチラン ノ ヒョウジ゛ ト インサツ
130 ' save "telno6.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 ED1$="### &                    &"
180 ED2$=" TEL   &                  &"
190 '
200 LOCATE 24,0 : COLOR 4
210 PRINT " デ゛ンワ バ゛ンゴ゛ウ イチラン ノ ヒョウジ゛ ト インサツ"
220 COLOR 0 : LOCATE 20,1
230 PRINT STRING$(40,149)
240 LOCATE 20,9
250 PRINT STRING$(40,149)
255 GOSUB *MENU
260 LOCATE 30,10 : COLOR 6
270 PRINT "  プ゜リンタ ヲ オモチデ゛スカ ? "
280 COLOR 0 : LOCATE 30,12
290 PRINT "Y・・・・ [,]   N・・・・[.]"
300 I$=INKEY$ : IF I$="" THEN GOTO 300
310 IF I$="," THEN INSATSU=-1 : GOTO 330
320 IF I$<>"." THEN BEEP : BEEP : GOTO 300
```

```
330 LOCATE 20,10 : PRINT SPC(50)
340 LOCATE 20,12 : PRINT SPC(50)
350 '
360 GOSUB *YOMU1
370 DIM A$(C),B$(C) : C=0
380 GOSUB *YOMU2
390 '
395 IF NARABEKAE THEN GOSUB *SORT
396 '
400 LOCATE 26,10 : COLOR 4
410 PRINT SPC(6);"5 ﾒｲ ｽﾞﾂ ﾋｮｳｼﾞ ｼﾏｽ ";SPC(5)
420 COLOR 0 : LOCATE 30,12
430 PRINT "RET ｷｰ ﾃﾞ ﾂｷﾞ ﾉ ﾍﾟｰｼﾞ ﾍ "
440 '
450 GOSUB *SHUTSURYOKU
460 '
470 CLS
480 LOCATE 30,3
490 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
500 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
510 FOR I=1 TO 2000 : NEXT I
520 RUN "menu02"
530 '
1000 '─────────────────────────────
1010  *YOMU1
1020 '─────────────────────────────
1030 OPEN "2:telno1.dat" FOR INPUT AS #1
1040 WHILE NOT EOF(1)
1050   INPUT #1,A$,B$
1060   C=C+1
1070 WEND
1080 CLOSE #1
1090 RETURN
1100 '
2000 '─────────────────────────────
2010  *YOMU2
2020 '─────────────────────────────
2030 OPEN "2:telno1.dat" FOR INPUT AS #1
2040 WHILE NOT EOF(1)
2050   C=C+1
2060   INPUT #1,A$(C),B$(C)
2070 WEND
```

```
2080 CLOSE #1
2090 RETURN
2100 '
3000 '--------------------------------------------------------
3010  *SHUTSURYOKU
3020 '--------------------------------------------------------
3030 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
3040 FOR I=1 TO C STEP 5
3050   FOR J=1 TO 5
3060      M=I+J-1
3065      IF M>C THEN GOTO 3150
3070      LOCATE 20,2+J
3080      PRINT USING ED1$;M,A$(M);
3090      IF NOT INSATSU THEN GOTO 3110
3100      LPRINT USING ED1$;M,A$(M);
3110      PRINT USING ED2$;B$(M)
3120      IF NOT INSATSU THEN GOTO 3140
3130      LPRINT USING ED2$;B$(M)
3140   NEXT J
3150   I$=INKEY$ : IF I$="" THEN GOTO 3150
3160   FOR K=3 TO 7
3170      LOCATE 20,K : PRINT SPC(50)
3180   NEXT K
3190 NEXT I
3200 RETURN
3210 '
4000 '--------------------------------------------------------
4010  *MENU
4020 '--------------------------------------------------------
4030 LOCATE 30,3 : COLOR 6
4040 PRINT "     ﾒﾆｭｰ ｦ ﾄﾞｳｿﾞ     "
4050 LOCATE 32,5 : COLOR 0
4060 PRINT "ﾄｳﾛｸ ｼﾞｭﾝ･････1"
4070 LOCATE 32,6
4080 PRINT "50ｵﾝ ｼﾞｭﾝ･････2"
4090 I$=INKEY$ : IF I$="" THEN GOTO 4090
4100 IF I$="2" THEN NARABEKAE=-1 : GOTO 4120
4110 IF I$<>"1" THEN BEEP : GOTO 4090
4120 LOCATE 25,3 : PRINT SPC(50)
4130 LOCATE 25,5 : PRINT SPC(50)
4140 LOCATE 25,6 : PRINT SPC(50)
4150 RETURN
```

```
4160 '
5000 '───────────────────────────────────
5010  *SORT
5020 '───────────────────────────────────
5030 FOR I=1 TO C-1
5040 K=I
5050   FOR J=I+1 TO C
5060     IF A$(J)<A$(K) THEN K=J
5070   NEXT J
5080   SWAP A$(I),A$(K)
5090   SWAP B$(I),B$(K)
5100 NEXT I
5110 RETURN
```

第4章

ランダムファイル[I]

4・1　ランダムファイルについて

ランダムファイル

ランダムファイルとはいったいなんでしょうか。

私たちが前章までに勉強したシーケンシャルファイルは、あるデータを読み出したいといっても直接読み出すことはできませんでした。検索にしろ、修正にしろ、そのデータファイルに書き込まれているデータを、好き嫌いにかかわらず、初めから読み込んでいかなければなりませんでした。

要するに、長さのまちまちのデータが連続的に書き込まれていたわけですから、書き込まれたデータの並びには規則性がありません。

これに対して**ランダムファイルは、データが一定の決められた長さで書き込まれています**から、当然、データの書き込まれ方に規則性がでてきますので、どこに位置しているかということさえわかれば、**直接に読み出すことができる**のです。

右ページの図をごらんください。

これは、前章の電話帳の氏名・電話番号のシーケンシャルファイルとランダムファイルでの書き込まれ方を図式化したものです。

シーケンシャルファイルでは、各項目が、コンマで区切られているだけで、ビッシリつめこまれています。△がありますが、これは姓と名を分けるために必要な区切りです。

バイト
〈注〉79ページのキャラクタコード表を見ると、「A」というキャラクタのアスキーコードは16進数で４１Ｈですが、この２桁の16進数を１バイトといいます。

文字は１文字が１バイトで書き込まれますから（ただし濁点は１文字分ですから「ガ」などは２バイト必要ですが、説明上ここでは１バイトと考えます)、「トクガワ　イエヤス」は９バイト、「オダ　ノブナガ」は７バイトというように、長さは一定しておりません。

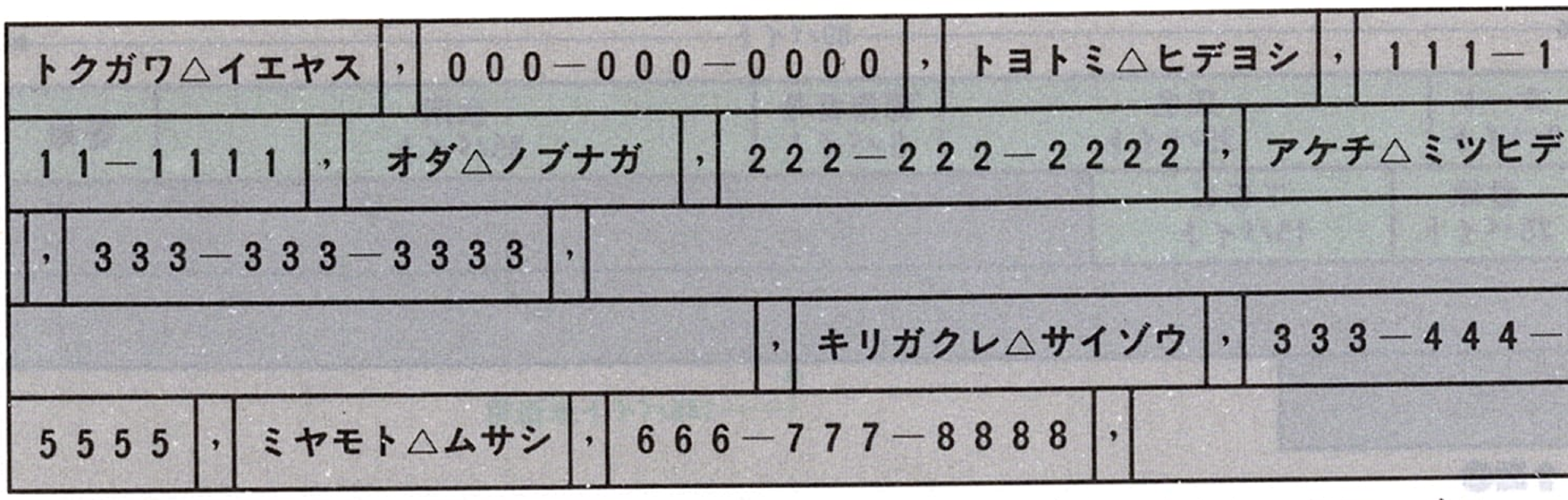
可変長レコード

このようにシーケンシャルファイルではレコードの長さが決まっていませんので**「可変長レコード」**と呼ばれ、制限もなく、ディスクの使用効率も非常によいのですが、ファイルの何番目に位置しているかは、一度データを全部読み出してみなければわかりません。

一方ランダムファイルは、図のように氏名および電話番号のうしろにムダなスペース（△印）がたくさんあいておりますが、いかにも整然と並んでいます。

図の場合ですと、どのデータも**氏名に17バイトが、電話番号に21バイトが割り当てられています。**シーケンシャルファイルでしたら「トクガワ　イエヤス」の場合、氏名と電話番号で９＋12で21バイトですむところをランダムファイルでは17バイトもよけいにムダな使い方をしているのです。

しかし、メリットもおおいにあります。**各データの書き込まれる長**



⬆シーケンシャルファイル

ランダムファイル⬇

トクガワ△イエヤス△△△△△△△△	000−000−0000△△△△△△△△△
トヨトミ△ヒデヨシ	111−111−1111
オダ△ノブナガ	222−222−2222
アケチ△ミツヒデ	333−333−3333
キリガクレ△サイゾウ	333−444−5555
ミヤモト△ムサシ	666−777−8888

さが一定していますので（ランダムファイルのレコードは「**固定長**」レコードといいます）、インデックスにより、**誰々さんは何番目に書き込まれているということがわかれば、**シーケンシャルファイルのように全部読み出さなくても**直接にその個所だけ読み出すことができます。**

固定長レコード

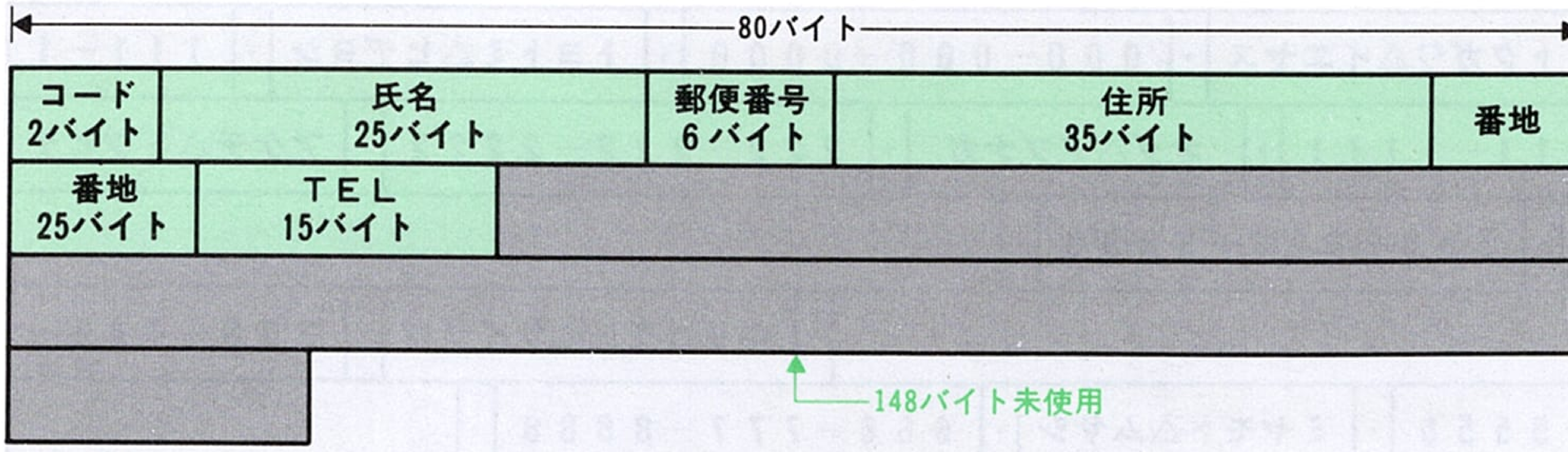
第２章第２節で簡単にふれましたが、ディスクにはサーフェス、トラック、セクタという単位がありました。**ランダムファイルでの読み書きは、**一番小さな単位、つまり**セクタごとに行われます。**そして１セクタは256バイトです。

セクタ

１セクタ＝256バイト

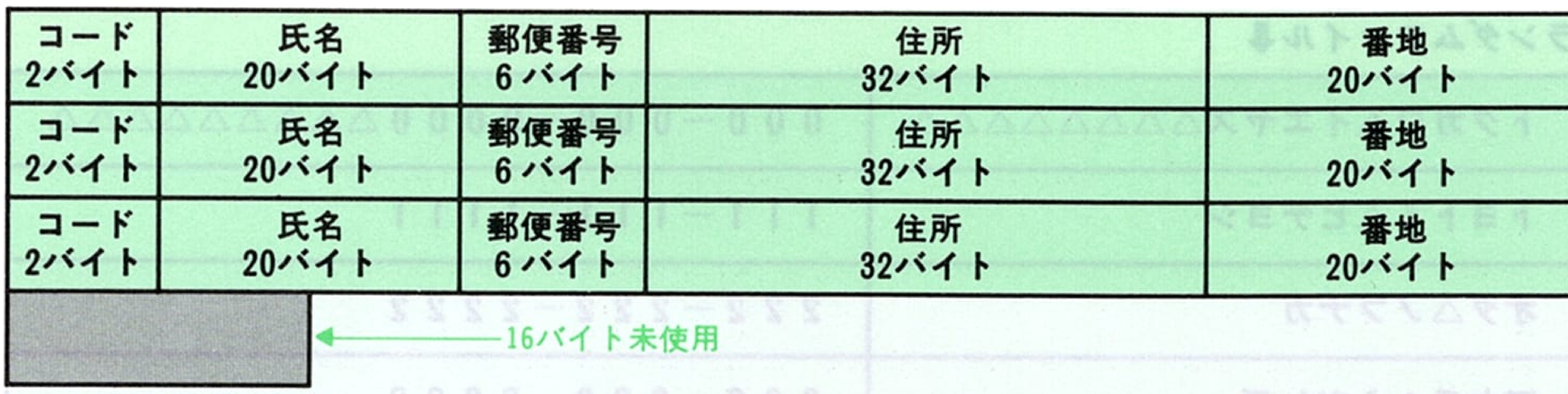
ですから、この256バイト内で各項目に決められた（固定された）長さを割り当てなければなりません。

次ページの図をごらんください。図❶、図❷とも、１セクタ256バイトにDM（ダイレクトメール）用の各項目を割り当てたものですが、図❶には１名分、図❷では３名分入っています。図❶の場合は、ずいぶん未使用の部分が多くムダな使い方をしていますし、一方図❷では



⬆図❶

⬇図❷



余りが16バイトで大変効率的です。

このくらいの人数ですと、メモリのムダづかいの様子はまだ実感としてわかりませんが、たとえば100人のDM（ダイレクトメール）用データファイルを作る場合、書き込みは256バイト（1セクタ）単位で行われますから、図❶の場合100セクタ必要になってきます。ところが図❷ですと、34セクタあれば十分です。66セクタも差がでてきます。

このように、1セクタ内でフォーマットをどう決めていくかによって、ランダムファイルにおいても、シーケンシャルファイルと同じようにディスクを有効に活用できるのです。

こう考えますと、慣れてしまえば、ランダムファイルのほうがずっと便利に使えますし、いや、**ビジネスの分野ではどうしてもランダムファイルでなければ用をなさない**といったほうがよいかもしれません。

以上、ランダムファイルについての概略を述べましたが、少しわかりにくかったかもしれませんが、わからなければそのままにして、少しずつ具体例を学習していきましょう。やがて、その仕組みが理解できると思います。

コンピュータは理屈、理論の面もおおいにありますが、とくにソフトのレベルでは、実際にやってみて、体で覚えてしまうのが一番早いようです。

4・2 ランダムファイルへの書き込み

それでは何かランダムファイルにデータを書き込んでみるところから始めましょう。

ファイルをOPENする

ランダムファイルの場合もシーケンシャルファイルと同じように、データファイルに何か書き込んだり、あるいは読み出したりするとき、ファイルをOPENしなければなりません。

これは、前にも説明しましたが、本とかノートの内容を見るときにページを開かなければならないのと同じです。

書式は、シーケンシャルファイルのように「モード」の指定はなく次のようになっています。

OPEN

```
OPEN "2:atena1.dat" as #1
```

- #1 ―― ファイル番号
- atena1.dat ―― ファイル名
- 2 ―― ドライブ番号

ファイル名の扱いはシーケンシャルファイルのときと同じです。**ドライブ1の場合にはドライブ番号は省略できます。**

ファイル番号（使用する**バッファ番号**）は、ディスク起動時の

ファイル番号

バッファ番号

```
How many files(0-15) ? ■
```

で指定した数を超えてはいけません。

また、シーケンシャルファイルでは「モード」の指定がありましたから、INPUTモード、APPENDモードの場合には、OPENしたときディスクにファイルがないと、

```
File not found
```

のエラーメッセージがでましたが、ランダムファイルの場合はOPEN文だけではエラーは出ません。

このOPEN文でファイルを開いてはじめて、データを読んだり書いたりすることが可能になります。

ファイルをCLOSEする

CLOSE

ファイルをOPENして、使い終ったらCLOSEすることにしましょう。これは、シーケンシャルファイルの場合とまったく同じです。

```
CLOSE #1,#2
```

#1, #2 ― ファイル番号

ファイル番号が省略されたときは、そのときOPENしているファイルがすべてCLOSEされます。ファイル番号は上記のように2つでも3つでもかまいません。コンマで区切って複数指定できます。

データ書き込みの前準備

ファイルをOPENしたということは、「そのファイルをこれから使いますよ」と宣言したことにはなりますが、このままでは書き込みはできません。前準備が必要なのです。

158ページでダイレクトメール用のデータの割り当てを行ったように、**1セクタ（256バイト）の中を、どんな項目でどんな割りつけをして使うのか、その形を決めておかなければいけないのです。**

氏名には何バイト、住所には何バイト、というように定義しておく必要があるのです。**この定義はFIELD（フィールド）文によって行います**が、ランダムファイルでは、この決められたフォーマットに従って、どのレコードも同じ様式で書き込まれます。

FIELD文

FIELD文の書式は次のとおりです。

```
FIELD #1,15 AS A1$,20 AS A2$,50 AS A3$
```

A1$, A2$, A3$ ― 文字変数
15, 20, 50 ― フィールド幅（バイト数）
#1 ― ファイル番号

この例ですと、ファイル番号1番（というより1番バッファ）に文字変数A1$、A2$、A3$にそれぞれ15バイト、20バイト、50バイトのフィールド幅を割り当てたことになります。ですから、**実質使用フィールド幅は85バイト**になり、**残り（256−85）の171バイトは使われない**ことになります。

フィールド幅の合計が256バイトになるまで、どのように割り当ててもかまいません。そして**割り当てを行う変数は必ず文字変数を使いま**

す（扱うデータが数値であっても、です）。

FIELD文によって、バッファ内に変数のエリアが決まりましたが、書き込む前にもう1つ準備が必要です。

それは、**このFIELD文の定義に従って、書き込むデータをバッファ上にセットしなければなりません。**つまり、FIELD文で使われている変数に、それぞれ対応するデータをはめこむのです。

この、データをバッファ上にセットする方法に次の2つがあります。

```
❶ LSET A1$=X$ ←—— バッファ上のA1$にX$の中身をセット
❷ LSET A2$=Y$ ←—— バッファ上のA2$にY$の中身をセット

❸ RSET A3$=Z$ ←—— バッファ上のA3$にZ$の中身をセット
```

❶も❷も❸もともにバッファ上の変数に、X＄、Y＄、Z＄に入っている文字列をセットすることに変わりないのですが、❶と❷の場合は、**LEFT**(レフト)、つまり**左づめにセット**され、❸の場合は**RIGHT**(ライト)、つまり**右づめにセット**される点が違います。いずれの場合も、セットしようとした文字列がフィールド幅より長いときは、余分な文字は無視されます。

LSET
●読み方は「エルセット」または「レフトセット」です。

RSET
●「アールセット」または「ライトセット」と読みます。

次に、フィールドは文字変数で定義されていますから、数値データをセットするときには、これを文字型に変換しなければなりません。数値が**整数**か、**単精度**か、**倍精度**かによって、次のように変換方法が異なります。

```
LSET B1$=MKI$(X%) ←—— 整数データの変換

LSET B2$=MKS$(Y!) ←—— 単精度データの変換

LSET B3$=MKD$(Z#) ←—— 倍精度データの変換
```

この場合LSETになっていますが、RSETでもかまいません。

また、MKIのMKは「MAKE」の略だと思われますので、それぞれ**「メイクインテジャードル」「メイクシングルドル」「メイクダブルドル」**と読むことにしましょう。

MKI＄
MKS＄
MKD＄
〈注〉MKI＄は「エム・ケー・アイ・ドル」と読んでもかまいません。

ここで1つ大切な注意があります。整数データ、単精度データ、倍精度データに対応するバッファ上の文字変数（上の例ではB１＄、B２＄、B３＄）はフィールド幅が決まっていて、それぞれ**2バイト**、**4バイト**、**8バイト**となっています。

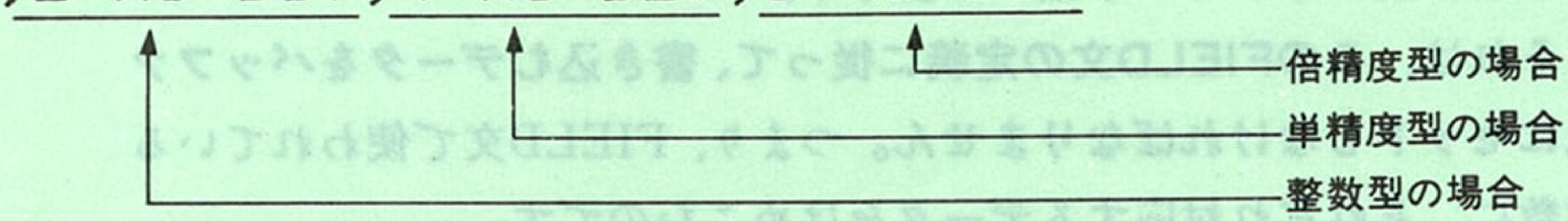
ですから、数値データを書き込むときは、数字の型によって**FIELD（フィールド）を定義**しておかなければなりません。

数値データのFIELD設定

FIELD文の書式

FIELD文の書式は次のとおりです。

```
FIELD #1,2 AS B1$,4 AS B2$,8 AS B3$
```



これでデータ書き込みの下準備ができました。

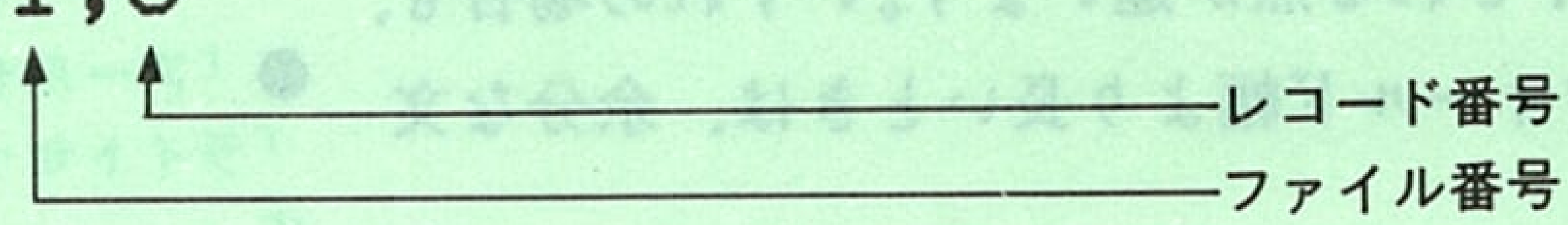
それではさっそく実践してみましょう。

データを書き込んでみよう

PUT文

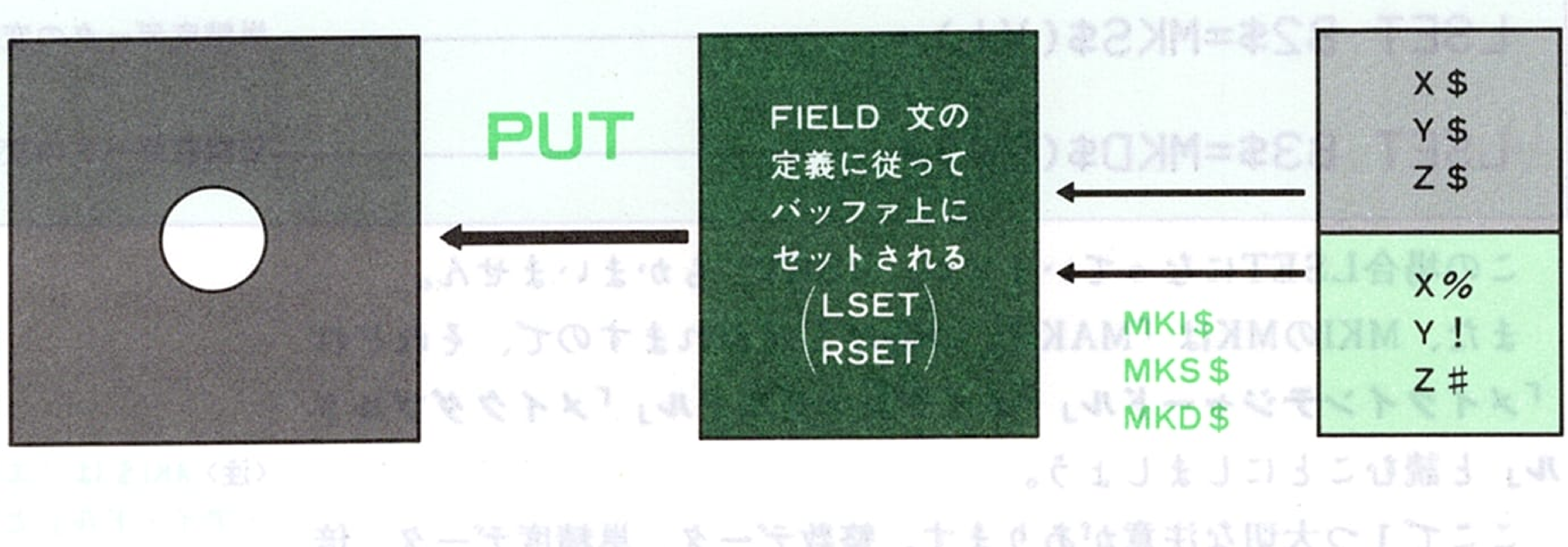
シーケンシャルファイルでは**WRITE＃（ライトシャープ）文**を使って書き込みをしましたが、ランダムファイルでは**PUT（プット）**を使います。書式は次のとおりです。

```
PUT #1,S
```



ファイル番号で指定されたバッファの内容（256バイト分）をSで示されるセクタ番号のところへ書き込む（PUTする）ということになります。レコード番号は省略することもできますが、その場合、直前に書き込みが行われたレコードの次にPUTされます。

これまでのプロセスをまとめると次の図のようになります。



MKI$、MKS$、MKD$によって文字列に変換された数値データ、および一般の文字変数が、FIELD文で決められたフィールド幅にそれぞれ**バッファ**上でセットされ（LSET、RSET）、このセットされたバッファの内容（256バイト、1セクタ分）が**PUT文**によってディスクに書き込まれます。

次に示すのは書き込みのテストプログラムです。

```
100 ' ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ
110 '
120 OPEN "2:test.dat" AS #1
130 FIELD #1,2 AS CD$,20 AS NM$,15 AS TL$
140 '
150 INPUT "ｺｰﾄﾞ(ｵﾜﾘ=9999)";CD
160 IF CD=9999 THEN GOTO 270
170 INPUT "ﾅﾏｴ ｦ ﾄﾞｳｿﾞ   ";A$
180 INPUT "ﾃﾞﾝﾜ ﾊﾞﾝｺﾞｳ ﾊ ";B$
190 S=S+1
200 '
210 LSET CD$=MKI$(CD)
220 LSET NM$=A$
230 LSET TL$=B$
240 PUT #1,S
250 '
260 GOTO 150
270 CLOSE #1
280 END
```

- 120 ← ファイルを開く
- 130 ← フィールドの設定
- 150〜180 ← データ入力
- 190 ← 1レコードのデータ入力ごとに、書き込むセクタ番号を1ずらす
- 210〜230 ← バッファ上にデータをセット
- 240 ← ディスクにデータを書き込む
- 260 ← ループ
- 270 ← ファイルを閉じる

プログラムは複雑ではありませんから、ごらんいただければわかると思いますが、書き込みまでの流れは次のようになっています。

データ書き込みの流れ

❶ **ファイルを開く（OPENする）**

❷ **フィールドを設定する（FIEDL文）**

❸ **データを入力する**

❹ **バッファ上にデータをセットする**

❺ **ディスクにデータを書き込む**

これが**データ書き込みの一般形**です。このパターンさえ覚えてしまえば、あとはディスクをどう有効利用するか、ファイルの形をどうするかなどの課題があるだけで、それほどむずかしくはありません。

コードは4桁の数値で入力することにしてありますので、210行でバッファ上にセットする場合、MKI$を使っています。

書き込むセクタ番号は、コード、名前、電話番号一式を入力するごとに1ずつカウントされ、ずれていきます。

このプログラムでは、コードで2バイト、名前で20バイト、電話番号で15バイトとっていて、これを1レコードとしていますから、全部で37バイト、したがって256から37を引いた残り219バイトは未使用のままです。

それではプログラムをRUNさせて、次の6名のデータを入力してみましょう。

1000……トクガワ　イエヤス……000−000−0000
1002……トヨトミ　ヒデヨシ……111−111−1111
8001……オダ　ノブナガ…………222−222−2222
5055……アケチ　ミツヒデ………333−333−3333
2008……アライ　ハクセキ………444−444−4444
7654……カイバラ　エッケン……555−555−5555

```
run
コード゛(オワリ=9999)? 1000
ナマエ ヲ ド゛ウゾ゛    ? トクガ゛ワ イエヤス
デ゛ンワ バ゛ンゴ゛ウ ハ ? 000-000-0000
コード゛(オワリ=9999)? 1002
ナマエ ヲ ド゛ウゾ゛    ? トヨトミ ヒデ゛ヨシ
デ゛ンワ バ゛ンゴ゛ウ ハ ? ■
```

上の画面は入力中の様子を示したものです。

コード、氏名、電話番号の1ブロックの入力が終るたびにカチッと音がしてディスクへの書き込みが行われます。6つのデータですから、最後のENDマーク（9999）が入力されるまでに、6レコード書き込まれたことになります。

入力が終りましたら、次の画面のステートメントをダイレクトで入力してみてください。いずれも、キーイン後リターンキーを押してください。

```
open "2:test.dat" as #1   ←いま書き込んだファイルをOPEN
Ok
print lof(1)   ←ファイルの長さを調べる
 6   ←セクタ数
Ok
close   ←必ず閉じることを忘れないでください
Ok
■
```

新しい関数がでてきました。**LOF（レングス・オブ・ファイル）関数**です。

これは、**ファイルの長さを知るための関数**で、PRINT文で前ページのように入力すると、すぐ下に長さを表示してくれます。前ページの画面では6となっていますね。

ちょっとわかりかけてきたのではありませんか？

そうです、私たちがいま入力したのは6人分のデータ、レコードにして6レコードだったのです。そしてこの場合、1レコードで256バイト、つまり1セクタを使っていましたから、データファイル「2：test. dat」分として全部で6セクタだけディスク上を占有したことになるのです。

次に、FILESコマンドでドライブ2側のディスクの内容を表示させて確認しておきましょう。

```
files 2
eitan1 dat 1  telno1 dat 1  test    dat 1
Ok
■
```

〈注〉この節で扱ったランダムファイルへの書き込みプログラム、および次節で扱う読み出しのプログラムはテスト用ですからセーブしなくてもかまいません。

LOF関数

● LOFをレングス・オブ・ファイルと読まず、「ロフ」と読んでもかまいません。

4・3 ランダムファイルのデータを読む

ランダムデータ

GET文

●GETは英語で「得る」という意味です。つまりディスクに書き込まれているデータを得るということ、いいかえれば「取り出す」ということですね。

さて、私たちは6名分のデータをランダムファイルに書き込むのに成功しました。でも、ほんとうに書き込まれているかどうか、ちょっと心配ですね。読み出して画面に表示してみましょう。

ランダムデータを読み出すには、GET(ゲット)文を使います。GETするのです。なんとなくわかるような感じがするでしょう？

そして書き込みのときにセクタ単位でPUTしたように、読み込むときもセクタ単位でGETすればよいのです。

書式は次のとおりです。

```
GET #1,S
```

（#1 ― ファイル番号、S ― セクタ番号）

この書式の意味は、**ファイル番号で指定されたランダムファイルのSセクタ目のデータを、ファイル番号に対応するバッファ上に読み出す**、ということです。

もう少し具体的にいいますと、上の例は現在OPENしている＃1のランダムファイルの第Sセクタを＃1の**バッファ**上に取り出すということになります。

バッファ

〈注〉本体メモリ上のある箇所に16個のバッファという領域が設けられています。その1つ1つが256バイトの広さを持っています。一方、ディスクもセクタ単位に分割されていて1セクタが256バイトの大きさですから、GET文で1セクタをGETして取り出すと、それがファイル番号で指定されたバッファに割り当てられます。

ファイル番号は1としてありますが、この数は0から15までの数を指定できます。ただし、How many files（0－15）？で入力した数を超えてはいけません。

突然GET文のことを書きましたので、中には、何もしなくてもすぐGETできるのか、と誤解する方もいるかもしれませんので補足しておきますが、データを読むときも書き込みのときと同じように、最初にファイルをOPENして次にフィールドを設定しておかなくてはなりません。

そして、このFIELD文によるフィールドの定義がなされていると、あるセクタをGETしたときに、自動的に、それぞれ定義されたフィールド幅をもつバッファ上の変数に割り当てられるのです。書き込みのときにはLSETまたはRSETを使ってセットしなければなりませんでしたが、読むときには必要ないのです。

このようにバッファ上に取り出され、バッファ上の変数に割り当てられたランダムデータを画面に表示するには、**文字型データはそのまま使えますが、数値データは書き込みの際に文字型に変換されている**

わけですから、もう一度数値に戻してやらなければなりません。

たとえば、B1＄、B2＄、B3＄というバッファ上の変数に、それぞれ2バイト、4バイト、8バイトの数値が文字型として代入されているとしますと、CVI、CVS、CVD*を用いて次のように数値に戻してやらなければなりません。

＊VAL関数も数値変換に使われますが、MKI＄などで文字列に変換してあるものは必ずCVI、CVS、CVDを用いて数値に戻さなければなりません。

```
X%=CVI(B1$) ←──── 整数値に変換
Y!=CVS(B2$) ←──── 単精度型数値に変換
Z#=CVD(B3$) ←──── 倍精度型数値に変換
```

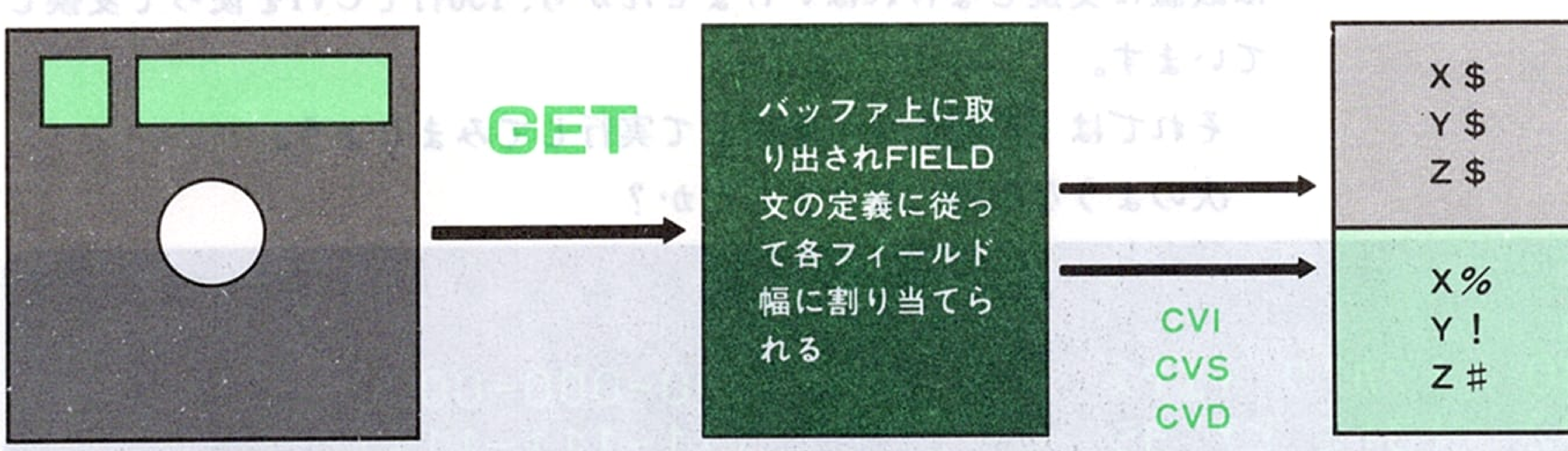
こうして数値型に変換しておきますと、N88-BASICモードで使っていたのと同じ扱いができるのです。

CVI
CVS
CVD

そして最後に、ファイルを使い終ったらCLOSEです。

〈注〉CVI、CVS、CVDはそれぞれコンバート・インテジャー、コンバート・シングル、コンバート・ダブルと読みます。あるいは単に、シーブイアイ、シーブイエス、シーブイデーでもかまいません。「コンバート」は「変換する」という意味です。

データを読む過程を図にすると次のようになります。



それではデータ読み出しを実行する前に、その手順を箇条書きにして整理しておきましょう。

❶ ランダムファイルを開く（OPENする）。
❷ FIELD文でフィールドを定義する。
❸ GET文を用いて1レコード（256バイト）ずつバッファ上に取りだす。
❹ 数値データの場合にはCVI、CVS、CVDを使って文字型を数値型に変換する。
❺ ランダムファイルを閉じる（CLOSE）する。

さあ準備完了です。先ほど書き込んだデータを読み出して画面に表示してみましょう。

プログラムを次に示します。

```
100 ′ ランダ゛ム ファイル ノ ヨミダ゛シ
110 OPEN "2:test.dat" AS #1 ←ファイルオープン
120 FIELD #1,2 AS CD$,20 AS NM$,15 AS TL$ ←フィールドの設定
130 FOR I=1 TO LOF(1) ←1セクタから最終セクタまでループ
140   GET #1,I
150   PRINT USING "####  ";CVI(CD$);  ┐
160   PRINT NM$;                      ├画面に表示
170   PRINT TL$                       ┘
180 NEXT I
190 CLOSE #1 ←ファイルを閉じる
200 END
```

130行のLOF(1)は、1番のファイルの長さ、つまり何セクタ使っているか、その値が与えられます。そして、その値の示す回数だけ140行のGET文によってディスク上のデータがバッファ上に取り出されます。これをPRINT文で画面に表示すればよいわけですが、CD$だけは数値に変換しなければいけませんから、150行でCVIを使って変換しています。

それでは f・5 キーを押して実行してみましょう。

次のような画面になりましたか？

```
run
1000  トクガ゛ワ イエヤス          000-000-0000
1002  トヨトミ ヒデ゛ヨシ          111-111-1111
8001  オダ゛ ノブ゛ナガ゛          222-222-2222
5055  アケチ ミツヒデ゛           333-333-3333
2008  アライ ハクセキ            444-444-4444
7654  カイバ゛ラ エッケン          555-555-5555
Ok
■
```

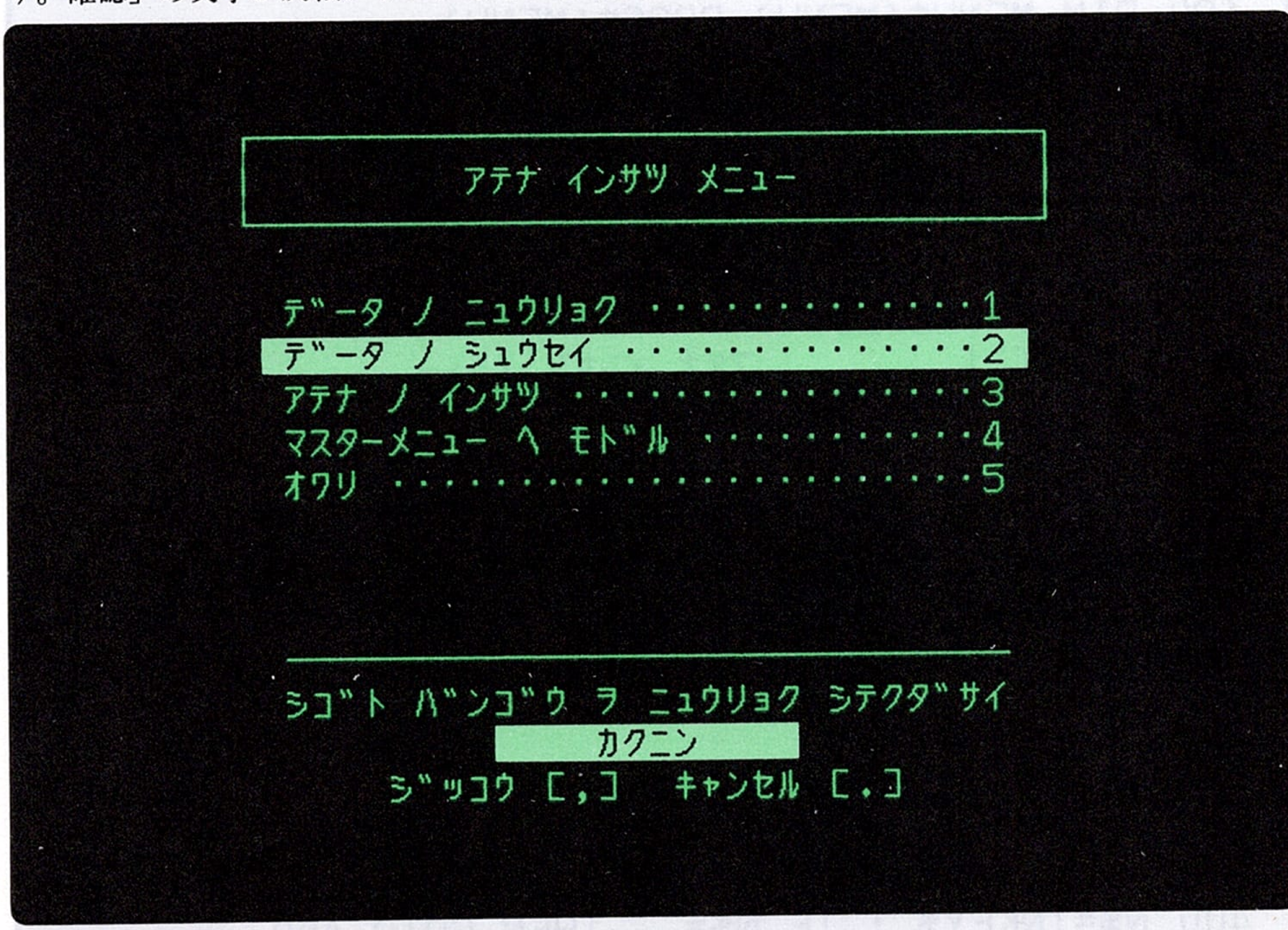
うまく読み出せたようです。私たちが前節で書き込んだ6人のデータが画面に表示されています。

4・4　メニュープログラムを作る

ランダムファイルの書き込み、読み出しが理解できたところで、今度はこれを使って簡単な宛て名印刷をテーマにプログラムを作っていきますが、最初にメニュープログラムを用意しておきましょう。

メニュー画面は次のようになります。

2番の「データの修正」を選択し、確認をうながしているところです。「確認」の文字が反転ブリンクしています。



プログラムリストを次ページに示しますが、82ページの

menu01

を一部修正することによって使えますので、これを流用することにします。まず、

```
load "menu01" ⏎
```

でプログラムをロードし、次ページのリストのコメントを入れた箇所をスクリーンエディタでなおしてください。

```
100 ' file name ････> menu03
110 ' 1985.2.15
120 ' ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ ﾌﾟﾛｸﾞﾗﾑ
130 ' save "menu03"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
230 '
240 PRINT TAB(20)
250 PRINT"┌──────────────────────────────┐"
260 PRINT TAB(20)
270 PRINT"│        ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ          │"
280 PRINT TAB(20)
290 PRINT"└──────────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"─")
380 LOCATE 22,13
390 PRINT "ｼｺﾞﾄ ﾊﾞﾝｺﾞｳ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)
420 IF N>MENU OR N=0 THEN BEEP : GOTO 400
430 '
440 COLOR 4
450 LOCATE 22,N+3
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "      ｶｸﾆﾝ      "
510 COLOR 0 : LOCATE 26,15
```

- menu03（行100）——ファイル名
- ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ ﾌﾟﾛｸﾞﾗﾑ（行120）——処理内容
- "menu03"（行130）——ファイル名
- ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ（行270）——見出しの変更
- OR N=0（行420）——追加

```
520 PRINT "ジッコウ [,]  キャンセル [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
610 '──────────────────────────────
620 '  menu data
630 '──────────────────────────────
640 DATA 5
650 DATA データ ノ ニュウリョク ··············1,atena1.bas
660 DATA データ ノ シュウセイ ···············2,atena2.bas
670 DATA アテナ ノ インサツ ·················3,atena3.bas
680 DATA マスターメニュー ヘ モドル ···········4,MASTER
690 DATA オワリ ··························5,dummy
```

●650～670行のDATA文を変更

なお、420行で

OR N=0

を追加しましたが、これを入れておかないと、メニュー番号入力のときに誤って [+] キーなどを押してしまっても確認のメッセージが表示されてしまいます。

修正できましたら、このプログラムを新しいファイル名でセーブしておきましょう。

```
save "menu03" ⏎
```

4·5　データの書き込み

項目

●右のように細かい分類の1つ1つを項目といい、項目の集まりをレコードといいます。

この節ではランダムファイルをディスク上に作りますが、このデータファイルには次の**項目**を記録しておくことにします。

❶ **コード番号**
❷ **氏名**
❸ **郵便番号**
❹ **住所**
❺ **番地**

これらを1レコードとして1セクタ内に書き込みます。少しムダな使い方ですが、これは学習のためですし、慣れるためにはある程度簡素化しておいたほうがよいと思うからです。

宛て名印刷の方法

宛て名印刷の方法は、宛て名印刷をしたい人のコードを次々と入力し、エンドマークを入れると、あとはプリンタがかってに印刷をしてくれるというふうにします。

コード番号には、いろいろなつけ方があります。下2桁を県別にするとか、職種の区別に第何桁を使うとか、また使用桁数もケースによってだいぶ違いがでてくると思います。しかし、この本では、データの入出力をどう扱うか、という点にテーマをしぼっていますので、コードのつけ方にはとらわれずに話を進めることにいたします。

入力画面

プログラムに入る前に画面の説明をしておきます。

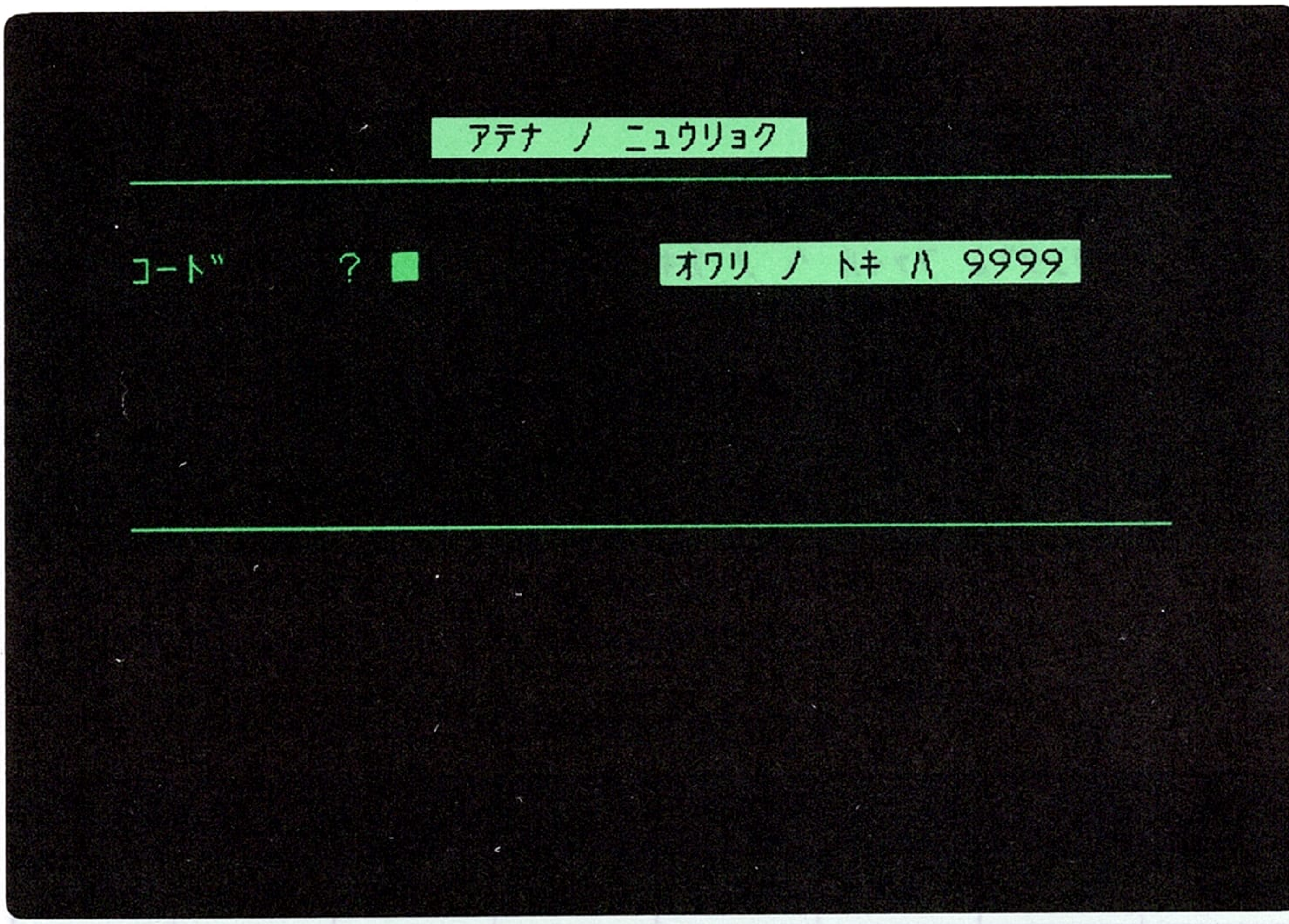
まずプログラムを実行すると、右ページ上の画面のようになりコード番号の入力を求めてきます。

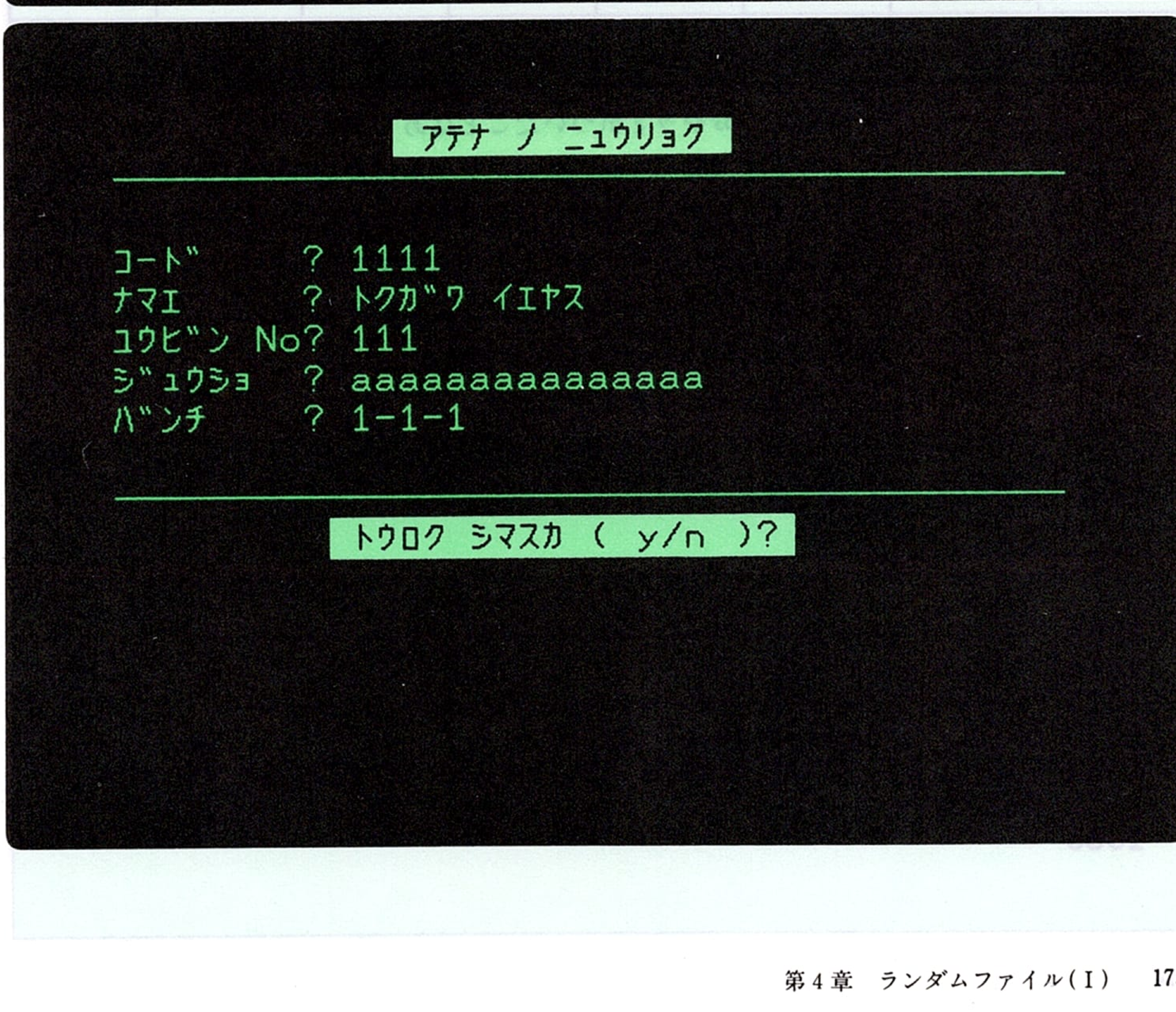
コード入力行の右端には**オペレーションメッセージ**が点滅しています。入力を終りたければ「9999」を入力します。

コードを入力したら、名前、郵便番号、住所、番地と、次々に入力し、全項目入力し終ると右ページ下の画面のようになります。

ここで間違いなければ登録することにして大文字でも小文字でもいいですからYをキーインします。すると、これらのデータがディスクに書き込まれ、再び右ページ上の画面になります。

なお、**「オワリ　ノ　トキ　ハ　9999」**のメッセージはコード入力が終ると自動的に消えるようにしてあります。


アテナ ノ ニュウリョク
コート゛ ?
オワリ ノ トキ ハ 9999


アテナ ノ ニュウリョク
コート゛ ? 1111
ナマエ ? トクカ゛ワ イエヤス
ユウヒ゛ン No? 111
シ゛ュウショ ? aaaaaaaaaaaaaaa
ハ゛ンチ ? 1-1-1
トウロク シマスカ (y/n)?

プログラムの先頭部分は、いつものようにファイル名、作成年月日を変えたりするだけでよいでしょう。

```
100 ' file name ････> atena1.bas
110 ' 1985.2.15
120 ' ランダ゛ム ファイル ヘ ノ カキコミ
130 ' save "atena1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
```

次にデータ書き込み用のランダムファイルをOPENしなければなりませんが、各項目が使用するフィールドを次のように設定することにします。

●フィールドの割り当て●

氏名		郵便番号		住所		番地	
変数	バイト	変数	バイト	変数	バイト	変数	バイト
NM$	20	PN$	6	AD$	35	BA$	20

ここで、おやッと気づいたことはありませんか？

そうです、158ページの図のファイルに書き込む項目にはコードも入っていましたが、この表には見当たりませんね。コードは書き込まないでいいのでしょうか？　この理由はすぐ説明します。

とにかくフィールドが定義できましたから、ランダムファイルをOPENする部分をコーディングしてみましょう。

これはサブルーチンにします。

```
180 GOSUB *FOPEN
```

```
1000 '―――――――――――――――――――――
1010  *FOPEN
1020 '―――――――――――――――――――――
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
```

180行のGOSUB文のラベル名は

FILE OPEN

から取ったものですが、このラベル名をFをつけずに単に

＊OPEN

とだけするとSyntax errorになります。というのはOPENが予約語だからです。

サブルーチンの説明は省略します。

ランダムファイルにおいては、**インデックス**という考え方をよく使います。本書には巻末に索引がついていますね。この索引と同じと考えていただいてよいと思います。

インデックス

ある用語を本文中から探そうとしても、ページをめくっただけではなかなか見つけることができません。ところが索引を調べれば、その用語にページ数がつけてありますから、すぐに探すことができますね。ランダムファイルで使うインデックスも同じことなのです。

たとえば、次の4名のデータを1人を1レコード（1セクタ）としてランダムファイルに書き込むとします。

コード	氏名	郵便番号	住所	番地
1111	トクガワ　イエヤス	111	aaaaaaaaaaaaaaa	1−1−1
2222	トヨトミ　ヒデヨシ	222	bbbbbbbbbbbbbbb	2−2−2
3333	オダ　ノブナガ	333	ccccccccccccccc	3−3−3
4444	アケチ　ミツヒデ	444	ddddddddddddddd	4−4−4

このとき、ランダムファイルの考え方では、「コード」「氏名」「郵便番号」「番地」、この5項目を1セクタ内にすべて書き込むのではなく、**コードだけはシーケンシャルファイルに書く**ことにしているのです。

そうしますと、この**シーケンシャルファイルに書き込まれたコードが、本でいう索引の役割をする**のです。

なぜかといいますと、シーケンシャルファイルは連続的に前のほうから順序よくデータがつめこまれていました。覚えていますか？

つまり、上の例でいいますと、「1111」は1番目に、「2222」は2番目に、「3333」は3番目に、「4444」は4番目に書き込まれます。と同時に、そのコードに対応する「トクガワ」「トヨトミ」「オダ」「アケチ」は、ランダムファイルの第1セクタ、第2セクタ、第3セクタ、第4

GET

セクタに書き込まれます。ということは、キーボードからコードを入力したとき、**そのコードがシーケンシャルファイルのX番目にあったとしますと、ランダムファイルのXセクタ目をGETすれば、希望する人のデータが得られる**わけです。

このことを図解したのが次の図です。

第1番目	1111
第2番目	2222
第3番目	3333
第4番目	4444

⬅シーケンシャルフファイル

たとえばコード番号3333の人のデータがほしければ、ランダムファイルの第3セクタをGETすればデータが得られる

GET

第1セクタ	トクガワ　イエヤス	111	aaaaaaaaaaaaaaa	1−1−1
第2セクタ	トヨトミ　ヒデヨシ	222	bbbbbbbbbbbbbbb	2−2−2
第3セクタ	オダ　ノブナガ	333	ccccccccccccccc	3−3−3
第4セクタ	アケチ　ミツヒデ	444	ddddddddddddddd	4−4−4

ご理解いただけたでしょうか？

要するに、データをランダムファイルにセーブするときに、インデックス用のシーケンシャルファイルも一緒に作っておくのです。

初回の書き込み、それとも追加？

さて、メニュープログラムで1番の「データの入力」を選択すると、この**「atena1.bas」**が実行されるわけですから、初回の書き込みも複数回目の書き込みも、いずれも考慮してコーディングしなければなりません。

OUTPUTモード

初回でしたら、インデックス用のシーケンシャルファイルは**OUTPUTモード**でよいですが、2回目、3回目にOUTPUTモードでオープンすると、それまでに作ったインデックスファイルが壊れてしまいます（OUTPUTは新規にファイルに書くときのモードでした）。

APPENDモード

それでは常に**APPENDモード**でOPENすればよいのでしょうか。一

見正しいように思えますが、そうではないのです。
たとえば、**初回の入力時にAPPENDモードでOPENすると、**

File not found

というエラーメッセージが返ってきます。
そこで、初回でも複数回目でも入力可能なようにするには、ちょっとした細工が必要なのです。次のステートメントを入れておきます。

```
190 IF LOF(1)<>0 THEN GOSUB *COUNT : GOTO 220
```

このLOF関数は、ランダムファイルをOPENしたときに、最大のレコード番号（あるいはそのファイルが使用しているセクタ数）を返してくるものです。そして、これは、ランダムファイルを初めてオープンしたときには0を与えます。

LOF関数
〈注〉この関数はファイルがオープンしているときにのみ使えます。レングス・オブ・ファイルの略で、ファイルの大きさをセクタ数で返してくれます。書き込みが行われていなければ当然ゼロを返します。

ですから、**LOFの値が0でなければすでにランダムファイルの書き込みは行われている**ということですし、と同時に、それはインデックス用のシーケンシャルファイルもすでにディスクの中に存在することを意味します。ということは、**この場合にはAPPENDモードでオープンすればよい**わけですね。
そして、ランダムファイルの書き込みは何セクタ目からすればよいかを決めるため、2000行からのサブルーチンへインデックスファイルを読みにいきます。
LOFが0ならば、新規にOUTPUTモードでインデックスファイルをOPENすることになります。
2000行からのサブルーチンは次のようになります。

```
2000 '―――――――――――――――――――――――――――
2010 *COUNT
2020 '―――――――――――――――――――――――――――
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   INPUT #2,DUMMY ←――――データのカラ読み
2060   REC=REC+1 ←――――――レコード数をカウントする
2070 WEND
2080 SEC=REC+1 ←――――――総データ数にプラス1した値が入力データを
                          書き込むランダムファイル上のレコード位置
                          になる
2090 CLOSE #2
2100 RETURN
```

2030行はインデックス用ファイルのオープンです。シーケンシャル

INPUT＃文

EOF関数

ファイルの読み出しは**INPUT＃（インプットシャープ）文**で行いますが、ファイルの終りのチェックは**EOF（エンド・オブ・ファイル）関数**を使います。

WHILE～WEND文

最後のデータを読み出すまで**WHILE～WEND文**でループさせればよいですね。

なお、ランダムファイルへの書き込み位置は（末尾のデータ数）プラス1になります。これは2080行で処理しています。

RETURN文によりサブルーチンから帰った場合は、220行からのAPPENDモードへ、そうでない場合は、初回のデータ入力ということになりますから、OUTPUTモードでインデックスファイルをOPENします。

```
200 OPEN "2:index1.dat" FOR OUTPUT AS #2
210 SEC=1 : GOTO *NYUURYOKU
220 OPEN "2:index1.dat" FOR APPEND AS #2
230 '
```

210行でランダムファイルへの書き込み位置を示す変数SECは1となっていますが、**この場合、初回の書き込みですから、ファイルの第1セクタからPUTしていきます。**

キーボードからのデータ入力

次はデータ入力部分ですが、173ページの画面をもう一度頭に描いてからお読みください。

270～300行は入力画面のレイアウトです。モノクロモードでのカラー文を使って見出しを反転表示させています。

データ入力を終りにしたいときはコードに対して「9999」をキーインします。この「キーオペレーションメッセージ」はコード以外の項目に対しては必要ないので、370行でこれを消去し、名前の入力へ移しています。

みなさんはミニコンとかオフコンという言葉をよく耳にすると思います。あるいはすでに会社・オフィスで導入されている方もいらっしゃるかもしれません。

これらのマシンはビジネス関係ですと普通BASICではなくCOBOL（コボル）という言語を使っていますが、このコボルにおいても私たちがやっていることと同じことを行っているのです。つまり、エラーメッセージ表示行とかデータ入力行というのがあって、ベーシックと同じような操作で行っているのです。

```
240 '─────────────────────────────
250  *NYUURYOKU
260 '─────────────────────────────
270 LOCATE 28,0 : COLOR 4
280 PRINT "    アテナ ノ ニュウリョク    " : COLOR 0
290 LOCATE 20,1 : PRINT STRING$(40,149)
300 LOCATE 20,9 : PRINT STRING$(40,149)
310 '
320 *LOOP
330 LOCATE 40,3 : COLOR 6
340 PRINT " オワリ ノ トキ ハ 9999  " : COLOR 0
350 LOCATE 20,3 : INPUT "コード     ";CD
360 IF CD=9999 THEN GOTO *OWARI
370 LOCATE 40,3 : PRINT SPC(20)
380 LOCATE 20,4 : INPUT "ナマエ     ";A1$
390 LOCATE 20,5 : INPUT "ユウビン No";A2$
400 LOCATE 20,6 : INPUT "ジュウショ  ";A3$
410 LOCATE 20,7 : INPUT "バンチ     ";A4$
420 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
430 LOCATE 28,10 : COLOR 6
440 PRINT "  トウロク シマスカ ( y/n )?"
450 COLOR 0
460 I$=INKEY$ : IF I$="" THEN GOTO 460
470 IF I$="Y" OR I$="y" THEN GOTO *KAKU
480 IF I$="N" OR I$="n" THEN GOTO *KESU
490 BEEP : BEEP : GOTO 460
500 '
```

5項目を入力し終ると、登録するかどうかの確認メッセージが最下行に表示されますから、これに「Y」または「N」で答えるようにします。これは**INKEY$**で受ければよいですね。

もし「**Y**」であれば「**書き込みルーチン**」へ、「**N**」の場合には一度「**データ入力部分を消去するルーチン**」へ飛ばして最初から入力のやりなおしをします。

490行は「Y」「N」以外のキーが入力された場合の措置です。

キーボードから入力したデータをディスクに書き込んだり、あるいは逆に読み出したりする場合、本体メモリ内の**バッファ**という場所を中継すると申し上げました。

このプログラムではランダムファイルに＃1を、シーケンシャルフ

ァイルに＃２を指定していました。

ここで１つのことを覚えてください。

バッファ番号

この＃１と＃２はバッファ番号を意味しているのです。ですから、このプログラムでは、

ランダムファイルは１番のバッファ

を、そして

シーケンシャルファイルは２番のバッファ

を使用していると言い換えることができます。

というわけで、１つのバッファを２つ以上のファイルで共用できないことがおわかりいただけると思います。

また、使用できるバッファ数は、システム立ち上げ時の

How many files (0−15)?

で入力した数だけ確保されます。リターンキーだけを押した場合はファイル数は「２」に初期設定されていますから注意してください。

ランダムファイルへの書き込み

登録の場合にはファイルへデータを書き込みます。次のプログラムを入力してください。

```
510 *KAKU
520 WRITE #2,CD      ←インデックスファイルにコードを書き込む
530 LSET NM$=A1$ ┐
540 LSET PN$=A2$ ├バッファ上にデータをセット
550 LSET AD$=A3$ │
560 LSET BA$=A4$ ┘
570 PUT #1,SEC       ←ランダムファイルにデータを書き込む
580 SEC=SEC+1        ←次の書き込みのためにセクタ数をふやす
590 '
```

WRITE文

シーケンシャルファイルへの書き込みは**WRITE文**を使います。

ランダムファイルの場合はまずバッファ上で、フィールド定義した変数にデータをセットし、そしてこのバッファ256バイト分のデータを**PUT文**を使ってディスクへ転送します。

PUT文

１レコード書き込んだら、次のレコード書き込みのためにセクタ番号を１つ進めておきます。

入力行の消去

入力データを登録して書き込みをしても、またキャンセルしても、いずれにしても現在画面に表示されているデータを消去し、もう一度新規入力に戻してやります。次のプログラムをキーインしましょう。

```
600 *KESU
610 LOCATE 0,3
620 FOR I=1 TO 5
630   PRINT TAB(20);SPC(40)
640 NEXT I
650 LOCATE 20,10 : PRINT SPC(40)
660 GOTO *LOOP
670 '
```

上から4行目（Y座標は3）から5行分消去します。

LOCATE文
TAB関数

まず**LOCATE文**で消去する行の先頭を決めておき、**TAB (20)** でカーソルを20字分飛ばし、そこからスペースを出力して文字を消します。

さらに、最下行の登録するかどうかのメッセージも消しておきましょう。

終りの処理

最後に、コードに「9999」が入力された場合の処理です。このときは、まずオープンしているファイルをすべてクローズし、画面をクリアしてからメニューへ戻るようにします。

〈注〉740行のタイマーは各自適当に調整してください。

```
680 '────────────────────
690  *OWARI
700 '────────────────────
710 CLOSE  : CLS
720 LOCATE 30,3
730 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
740 FOR I=1 TO 5000 : NEXT I
750 RUN "menu03"
760 '
```

プログラムの実行

さあ、これで入力が終りました。まずセーブしておきましょう。

```
save "atena1.bas" ⏎
```

セーブできたら、f・5 キーでRUNさせて次のデータを入力してみましょう。

コード	名 前	郵便番号	住 所	番地
1111	トクガワ イエヤス	111	aaaaaaaaaaaaaaa	1－1－1
2222	トヨトミ ヒデヨシ	222	bbbbbbbbbbbbbbb	2－2－2
3333	オダ ノブナガ	333	ccccccccccccccc	3－3－3
4444	アケチ ミツヒデ	444	ddddddddddddddd	4－4－4

うまくいきましたか？

成功したら次の節へ進みましょう。

〈注〉ファイルをクローズしてないとKILLコマンドは使えません。オープンしたまま削除しようとすると File already open のエラーになります。

また、**途中でエラーがでてストップしたら、ダイレクトモードで一度CLOSEしてから、チェックしてください。**バグ（プログラムの間違い）をとってから再度RUNする前に、すでにデータファイルがディスクに作られていたら、次のようにそれを**KILL（キル）コマンド**で削除してから始めましょう。正常に動くまでは、めんどうでも、こういったくりかえしが必要です。

```
kill "2:index1.dat" ⏎
kill "2:atena1.dat" ⏎
```

本節でのプログラムリストを次ページにかかげます。

〈注〉本節のプログラムでデータの書き込みをする場合、書き込み件数が17件以上の場合、書き込むセクタ番号は17，18，19……となり、たとえば5インチ両面倍密度ディスクに割り当てられたセクタ番号1～16を超えてしまい感覚的なずれが生じますが、たとえば17セクタ目に書き込む場合にはコンピュータ内部で「次のトラックの第1セクタ目」というふうに自動的に換算してくれますので、みなさんは深く考えないほうがよいでしょう。

```
100 ' file name ････> atena1.bas
110 ' 1985.2.15
120 ' ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ﾍ ﾉ ｶｷｺﾐ
130 ' save "atena1.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 GOSUB *FOPEN
190 IF LOF(1)<>0 THEN GOSUB *COUNT : GOTO 220
200 OPEN "2:index1.dat" FOR OUTPUT AS #2
210 SEC=1 : GOTO *NYUURYOKU
220 OPEN "2:index1.dat" FOR APPEND AS #2
230 '
240 '──────────────────────────────────────────
250  *NYUURYOKU
260 '──────────────────────────────────────────
270 LOCATE 28,0 : COLOR 4
280 PRINT "      ｱﾃﾅ ﾉ ﾆｭｳﾘｮｸ       " : COLOR 0
290 LOCATE 20,1 : PRINT STRING$(40,149)
300 LOCATE 20,9 : PRINT STRING$(40,149)
310 '
320 *LOOP
330 LOCATE 40,3 : COLOR 6
340 PRINT " ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999   " : COLOR 0
350 LOCATE 20,3 : INPUT "ｺｰﾄﾞ     ";CD
360 IF CD=9999 THEN GOTO *OWARI
370 LOCATE 40,3 : PRINT SPC(20)
380 LOCATE 20,4 : INPUT "ﾅﾏｴ      ";A1$
390 LOCATE 20,5 : INPUT "ﾕｳﾋﾞﾝ No";A2$
400 LOCATE 20,6 : INPUT "ｼﾞｭｳｼｮ   ";A3$
410 LOCATE 20,7 : INPUT "ﾊﾞﾝﾁ     ";A4$
420 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
430 LOCATE 28,10 : COLOR 6
440 PRINT "   ﾄｳﾛｸ ｼﾏｽｶ ( y/n )?  "
450 COLOR 0
460 I$=INKEY$ : IF I$="" THEN GOTO 460
470 IF I$="Y" OR I$="y" THEN GOTO *KAKU
480 IF I$="N" OR I$="n" THEN GOTO *KESU
490 BEEP : BEEP : GOTO 460
500 '
510 *KAKU
520 WRITE #2,CD
530 LSET NM$=A1$
540 LSET PN$=A2$
550 LSET AD$=A3$
560 LSET BA$=A4$
570 PUT #1,SEC
580 SEC=SEC+1
```

```
590 '
600 *KESU
610 LOCATE 0,3
620 FOR I=1 TO 5
630   PRINT TAB(20);SPC(40)
640 NEXT I
650 LOCATE 20,10 : PRINT SPC(40)
660 GOTO *LOOP
670 '
680 '────────────────────────────────────
690  *OWARI
700 '────────────────────────────────────
710 CLOSE  : CLS
720 LOCATE 30,3
730 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
740 FOR I=1 TO 5000 : NEXT I
750 RUN "menu03"
760 '
1000 '────────────────────────────────────
1010  *FOPEN
1020 '────────────────────────────────────
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
2000 '────────────────────────────────────
2010  *COUNT
2020 '────────────────────────────────────
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   INPUT #2,DUMMY
2060   REC=REC+1
2070 WEND
2080 SEC=REC+1
2090 CLOSE #2
2100 RETURN
```

4・6　データの修正

住所の変更とか、入力ミスにあとで気づいたとかした場合、プログラムさえしっかり作っておけば、データの修正は簡単です。

シーケンシャルファイルのときは、氏名の一部を入力して、それと同じ文字列をファイルの中から探して修正を加えていました。カタカナでの入力はちょっとめんどうですし、すでに書き込みずみのデータをすべて配列の中にとり込んでおかなければいけませんでした。

ところがランダムファイルとなると、キー入力はきわめて簡単。**数字列のコードをテンキーで入力すればよいのですし、コードは氏名と1対1の対応をなしていますから、同姓の人を考慮する必要がありません。**ただ、コード体系を完成させるまでは少々大変だと思いますが、あとで楽をするためには、初めに苦労はつきものですね。

それに、ディスクに保存できるデータ量もシーケンシャルファイルと比べて格段の差がでてきます。シーケンシャルファイルではすべてのデータを配列に読み込まなければならないので、コンピュータのメモリを考慮すれば、そこにおのずと制限がでてきます。

しかし、ランダムファイルになりますと、シーケンシャルに書き込まれているインデックスは配列にとり込む必要がありますが、**その他のデータの入出力は、すべて256バイトのバッファだけで事がすんでしまいます。**大変な違いですね。

たとえばMR用の5インチ高密度ディスクは片面が77トラックで構成され、これが表裏の2面ありますから全部で77×2＝154トラックあります。そしてこのディスクを**ユーザディスク**（フォーマットだけしてありシステムのコピーのないもの）として使う場合、サーフェス0の0番、35番トラックはファイル管理のために使われますので除外しなければなりませんから結局**154－2＝152トラック**使えます。

1トラック＝26セクタですから、**152×26＝3952セクタ**となります。

この章では1セクタに1人のデータしか書き込んでいませんが、もし3人分書き込むとすれば、

3952(セクタ)×3(人)＝11,856(人)

の資料をストックできるのです。

実用面でデータを管理するには、やはりランダム的な思考法が要求されるのです。

ユーザディスク
●市販の新しいディスクにフォーマッティング（初期化）だけを行って使える状態にし、システムなどのコピーをしないディスクをユーザディスクといいます。とくにデータファイル用として使ったりしますのでデータディスクとも呼ばれます。

それではコーディングにかかりますが、プログラムの流れを画面イメージで説明しておきましょう。

まず、プログラムを実行すると下に示す画面のようになりコード番号の入力を求めてきます。終了にしたければ、ここで**「9999」**をキーインします。

コードを入力してリターンキーを押し、もしそのコードが見つからないときには

トウロク　シテ　アリマセン!!

のエラーメッセージが表示されますし、見つかれば右ページ上の画面のように修正作業を行うことになります。

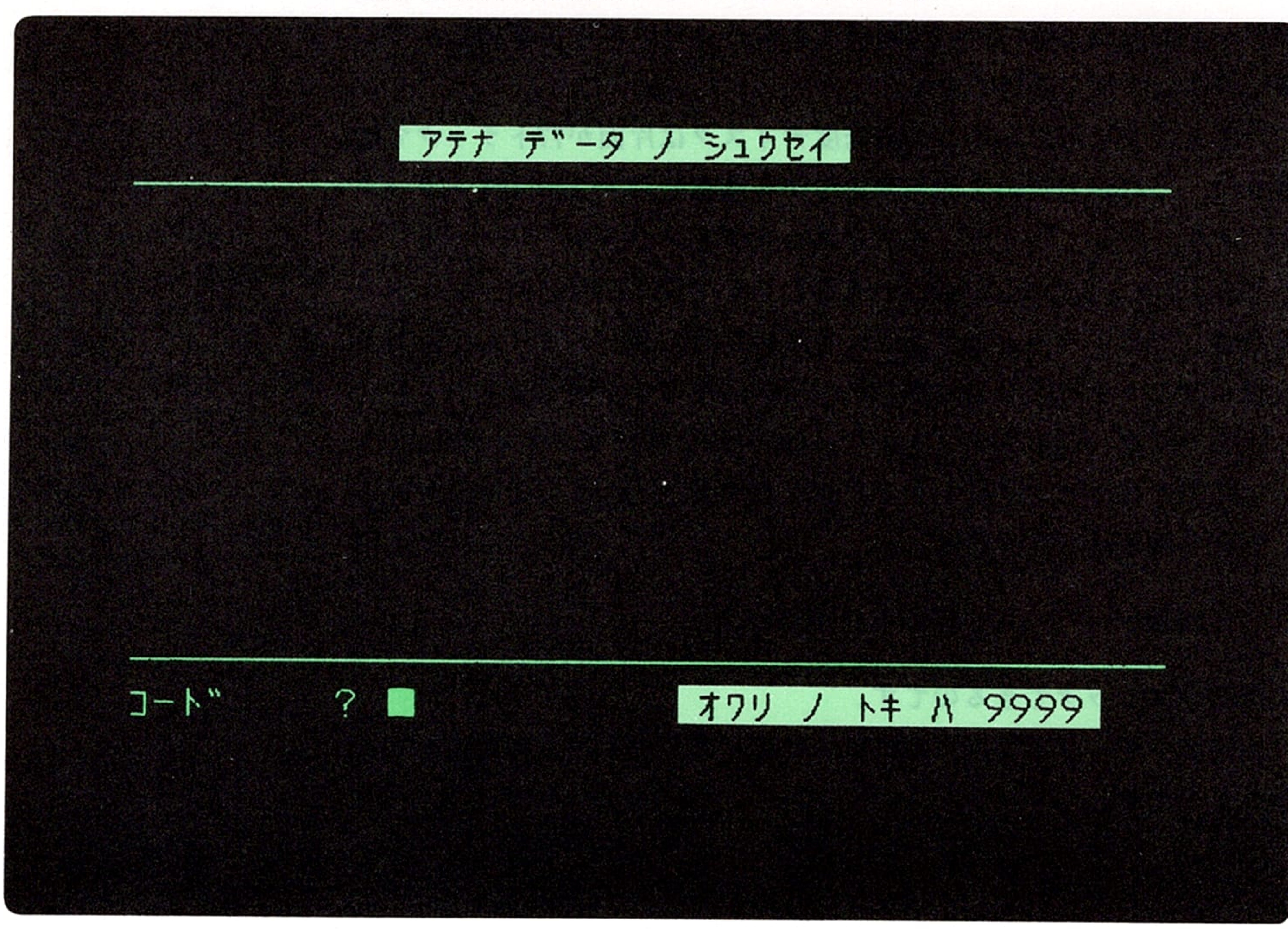
修正方法は、すでにディスクに書き込まれている氏名、郵便番号、住所、番地が表示されますから、修正しない項目にはリターンキーを押してパスし、修正項目はキーボードから入力します。

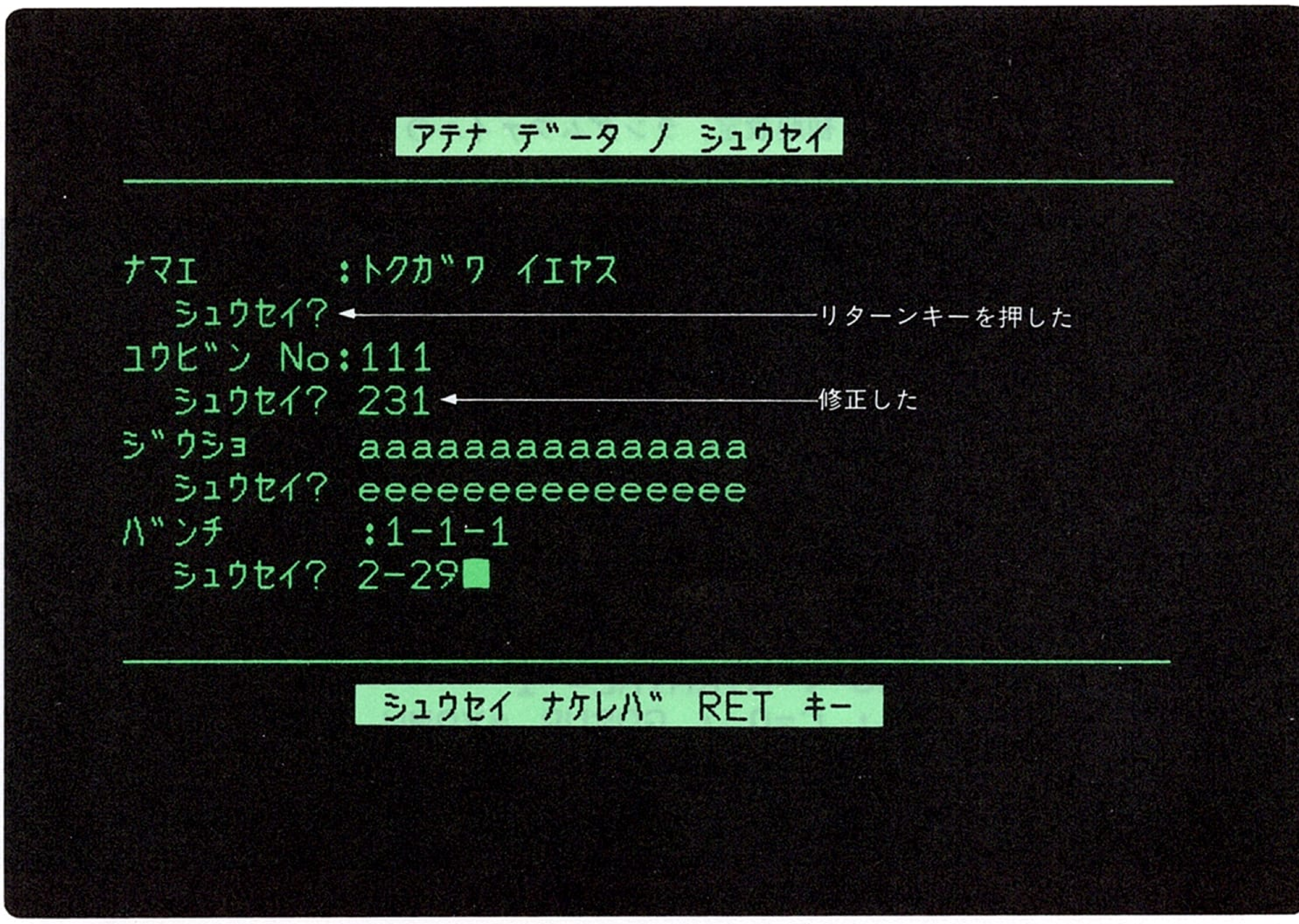
4項目の修正が終ると右ページ下の画面のようになり、登録するかどうか、つまり修正に間違いがないかどうかたずねてきます。

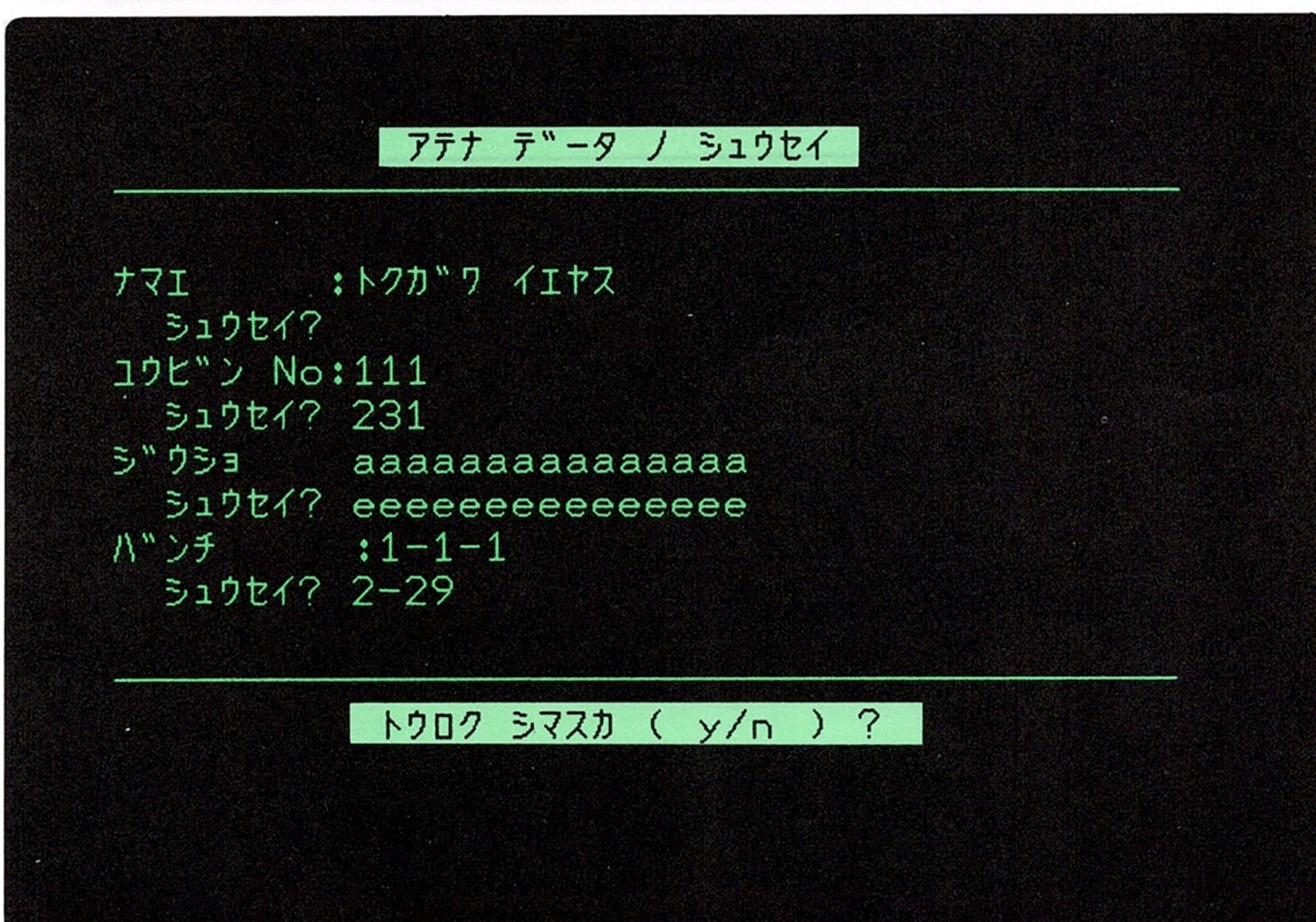
〈注〉確認メッセージに対してのYまたはNは大文字でも小文字でも機能します。

もう一度やりなおしたければ「N」を押して初めからやりなおしますし、それでよければ「Y」を入力してディスクに新しいデータを書き込みます。

修正作業終了後はメニューへ戻ります。




アテナ データ ノ シュウセイ
ナマエ :トクガワ イエヤス
シュウセイ?
リターンキーを押した
ユウビン No:111
シュウセイ? 231
修正した
ジュウショ aaaaaaaaaaaaaaa
シュウセイ? eeeeeeeeeeeeeee
バンチ :1-1-1
シュウセイ? 2-29
シュウセイ ナケレバ RET キー


アテナ データ ノ シュウセイ
ナマエ :トクガワ イエヤス
シュウセイ?
ユウビン No:111
シュウセイ? 231
ジュウショ aaaaaaaaaaaaaaa
シュウセイ? eeeeeeeeeeeeeee
バンチ :1-1-1
シュウセイ? 2-29
トウロク シマスカ (y/n) ?

プログラムの作成

まず、初期設定、ランダムファイルのオープン、そしてインデックス用ファイルのデータを配列に読み込むまでです。

```
100 ' file name ····> atena2.bas
110 ' 1985.2.15
120 ' ランダム データ ノ シュウセイ
130 ' save "atena2.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C) : C=0 : GOSUB *YOMU
210 '
```

サブルーチンも続けて入力しましょう。

```
1000 '————————————————————————————————————————
1010  *FOPEN
1020 '————————————————————————————————————————
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
2000 '————————————————————————————————————————
2010  *YOMU
2020 '————————————————————————————————————————
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
```

ランダムファイルのオープンは問題ないと思いますが、シーケンシャルファイルの読み出しについて説明しておきましょう。

190行でまず読み出しのサブルーチンへGOSUB文でジャンプします。

このときのジャンプはまだ1回目ですからNIKAIMEという変数の内容は0です。つまり、2060行のIF文の条件は**「偽」**ですから2080行へはジャンプせず2070行が実行され、**シーケンシャルファイルのカラ読み**が行われます。

〈注〉NIKAIMEはフラグ用に作った変数で「2回目」をローマ字にしたものです。XでもYでも、何を使ってもかまいません。

カラ読みがファイル末尾まで行われジャンプ先へ戻った時点で、

〈注〉「真」「偽」の説明は135ページに詳しくしてあります。

NIKAIME=−1

つまり、今度ジャンプするときは2回目ですので変数NIKAIMEを**「真」**にします。

200行で配列を宣言し、カウンタCをゼロクリアしてからシーケンシャルファイル読み出しのサブルーチンへ2回目のジャンプをします。サブルーチンでは1回目と同じようにシーケンシャルファイルをオープンしてデータを読み出すわけですが、今度は2060行のIF文の条件が「真」ですから2080行へ飛ぶことになり、配列CDにコードが読み込まれます。

前章では、いま説明した操作を行うのにカラ読み用サブルーチンと配列への読み込み用サブルーチンの2つを作って行っていましたが、このように1つのサブルーチンで処理することができるのです。

入力画面を作る

いつものように、処理名の表示と入力部分のレイアウトを決めておきましょう。

次を入力してください。

```
220 LOCATE 28,0 : COLOR 4
230 PRINT "   ｱﾃﾅ ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ    " : COLOR 0
240 LOCATE 20,1  : PRINT STRING$(40,149)
250 LOCATE 20,12 : PRINT STRING$(40,149)
260 '
```

データ修正のループ

この部分のプログラムを次ページに示します。

まず、310行で終りのときのキー操作を反転ブリンクして表示させ、320行でコード番号の入力です。

ここで「9999」が入力されれば修正作業は終りですので、画面をクリアしてメニューへ戻るルーチンへ飛ばします。

```
270 '
280 *LOOP
290 '
300 LOCATE 40,13 : COLOR 6
310 PRINT "   ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999 " : COLOR 0
320 LOCATE 20,13 : INPUT "ｺｰﾄﾞ      ";CD   ←コード入力
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)   ←配列内の該当コードを探す
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI
360 SEC=BANGOO
370 GOSUB *SHUUSEI   ←修正サブルーチンへ
380 GOTO *LOOP
390 '
```

SEARCH関数

コードが入力されると**SEARCH（サーチ）関数**を使って配列CDの中に入力されたコードがあるかどうか探します。

SEARCH関数の書式は次のとおりです。

SEARCH（整数配列名，データ）

これは整数配列の中から、指定された整数データがあるかどうか探し出す関数で、もし見つかればその要素番号を返し、見つからなければ－1を値として返します。

とくに注意することは、**この関数は整数配列に対してのみ有効で**あり、もしプログラムの先頭で

DEFINT A-Z

などのように変数の整数型宣言がなされていないときは、配列名の後ろに**%（パーセントマーク）**をつけて

SEARCH (CD%,CD)

などとしてください。

さて、配列CDの中をコードCDがあるかどうかSEARCH関数を使ってチェックし、その値を変数BANGOOに代入します。そしてこの値がもし－1であれば該当コードがなかったことになりますので、サブルーチンへジャンプさせ、

トウロク　シテ　アリマセン!!

のエラーメッセージを表示させます。

もし見つかれば、その値を変数SECに代入しなおし修正サブルーチンへジャンプさせます。SECへの代入はしなくてもよいのですが、変数BANGOOの内容が配列内の要素番号になっており、さらにこれはデータの登録順番を示しているのでランダムファイルはこの値で示されるセクタをGETすればよく、一方GET文では

GET #1,SEC

などのように明らかにセクタをGETしているのだとわかるような変数を使ったほうがわかりやすいのでこのようにしました。

見つからないとき

次に示すサブルーチンで処理します。

```
3000 '――――――――――――――――――――――――――――――
3010  *MITSUKARANAI
3020 '――――――――――――――――――――――――――――――
3030 LOCATE 20,13 : PRINT SPC(40)
3040 LOCATE 30,13 : COLOR 6
3050 PRINT " ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,13 : PRINT SPC(40)
3090 RETURN *LOOP
3100 '
```

最下行に少しの間エラーメッセージを表示させてから戻りますが、3090行の**RETURN文**のように、単にRETURNとするだけでなく、戻り先を指定できます。指定先は行番号でもラベルでもかまいません。覚えておくと便利です。 RETURN文

データ修正サブルーチン

このサブルーチンを次ページに示します。

修正の原理は簡単です。4030行で該当セクタをバッファ上に取り出し、それを画面に表示させ、修正があれば正しいデータをキーボードから入力し、修正がなければリターンキーを押すだけです。リターンキーが押されたときは、たとえば4090行の場合Ａ１＄にはヌルストリ

```
4000 '―――――――――――――――――――――――――
4010  *SHUUSEI
4020 '―――――――――――――――――――――――――
4030 GET #1,SEC
4040 LOCATE 20,13 : PRINT SPC(40)
4050 COLOR 4 : LOCATE 28,13
4060 PRINT "  ｼｭｳｾｲ ﾅｹﾚﾊﾞ RET ｷｰ    "
4070 COLOR 0
4080 LOCATE 20,3 : PRINT "ﾅﾏｴ       :";NM$
4090 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A1$
4100 IF A1$="" THEN A1$=NM$
4110 PRINT TAB(20);"ﾕｳﾋﾞﾝ No:";PN$
4120 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A2$
4130 IF A2$="" THEN A2$=PN$
4140 PRINT TAB(20);"ｼﾞｳｼｮ      ";AD$
4150 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A3$
4160 IF A3$="" THEN A3$=AD$
4170 PRINT TAB(20);"ﾊﾞﾝﾁ      :";BA$
4180 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A4$
4190 IF A4$="" THEN A4$=BA$
4200 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
4210 COLOR 6 : LOCATE 28,13
4220 PRINT "  ﾄｳﾛｸ ｼﾏｽｶ ( y/n ) ? "
4230 COLOR 0
4240 I$=INKEY$ : IF I$="" THEN GOTO 4240
4250 IF I$="Y" OR I$="y" THEN GOTO *KAKU
4260 IF I$="N" OR I$="n" THEN GOTO *KESU
4270 BEEP : BEEP : GOTO 4240
4280 '
4290 *KAKU
4300 LSET NM$=A1$ : LSET PN$=A2$
4310 LSET AD$=A3$ : LSET BA$=A4$
4320 PUT #1,SEC
4330 KAERU=-1
4340 '
4350 *KESU
4360 LOCATE 0,3
4370 FOR I=1 TO 8
4380   PRINT TAB(20);SPC(40)
4390 NEXT I
4400 LOCATE 20,13 : PRINT SPC(40)
4410 '
4420 IF KAERU THEN KAERU=0 : RETURN
4430 GOTO 4040
```

ングが入りますから、4100行の

IF A1$="" THEN A1$=NM$

ダブルコーテーションの間にスペースを入れないでください。2つ続けてダブルコーテーションを打つのです。

にひっかかり、それまでのデータがそのまま維持されます。修正したときは当然、A1$の内容は新しいデータになります。

4項目に対しての処理がすむと、登録するかどうかの確認です。

4240行のINKEY$によるキー入力待ちに対して「Y」ならディスクへの書き込みへ、「N」なら入力画面を消してもう一度修正のやりなおしです。

ひとつ注意することは、ディスクへ書き込んだあと4330行で

KAERU=-1

と、このサブルーチンから帰るかどうかのチェック用変数KAERUを「真」にしていることです。

そして入力画面を消去してから、もしKAERUが「真」なら4420行にひっかかりサブルーチンからリターン、そうでなければ4040行へ飛んでもう一度修正のやりなおしです。

終りの処理

最後に、コードに対して「9999」が入力されたときのステートメントをキーインしましょう。

```
400 '―――――――――――――――――――――――――――――――
410  *OWARI
420 '―――――――――――――――――――――――――――――――
430 CLOSE  : CLS
440 LOCATE 25,3
450 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
460 FOR I=1 TO 2000 : NEXT I
470 RUN "menu03"
480 '
```

これでプログラムは終りです。忘れずにセーブしましょう。

```
save "atena2.bas" ⏎
```

プログラムの実行

セーブできましたら f・5 キーで実行してみてください。

コード番号を聞いてきますから、みなさんの好きなコードを入力しメッセージに従って修正してみましょう。1つか2つのデータを修正したらコードに「9999」を入力しメニューへ戻り、ここで2番の「データの修正」を選択し、このプログラムが実行されたら先ほど修正したデータのコードをキーインし、修正が正しく行われているかどうかチェックしてください。

次にプログラムリストをかかげます。

```
100 ' file name ････> atena2.bas
110 ' 1985.2.15
120 ' ﾗﾝﾀﾞﾑ ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ
130 ' save "atena2.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C) : C=0 : GOSUB *YOMU
210 '
220 LOCATE 28,0 : COLOR 4
230 PRINT "   ｱﾃﾅ ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ    " : COLOR 0
240 LOCATE 20,1  : PRINT STRING$(40,149)
250 LOCATE 20,12 : PRINT STRING$(40,149)
260 '
270 '―――――――――――――――――――――――――――――――――
280 *LOOP
290 '―――――――――――――――――――――――――――――――――
300 LOCATE 40,13 : COLOR 6
310 PRINT "  ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999 " : COLOR 0
320 LOCATE 20,13 : INPUT "ｺｰﾄﾞ     ";CD
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI
360 SEC=BANGOO
370 GOSUB *SHUUSEI
380 GOTO *LOOP
390 '
400 '―――――――――――――――――――――――――――――――――
410  *OWARI
420 '―――――――――――――――――――――――――――――――――
430 CLOSE  : CLS
```

```
440 LOCATE 25,3
450 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
460 FOR I=1 TO 2000 : NEXT I
470 RUN "menu03"
480 '
1000 '--------------------------------------------
1010  *FOPEN
1020 '--------------------------------------------
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
2000 '--------------------------------------------
2010  *YOMU
2020 '--------------------------------------------
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
3000 '--------------------------------------------
3010  *MITSUKARANAI
3020 '--------------------------------------------
3030 LOCATE 20,13 : PRINT SPC(40)
3040 LOCATE 30,13 : COLOR 6
3050 PRINT " ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,13 : PRINT SPC(40)
3090 RETURN *LOOP
3100 '
4000 '--------------------------------------------
4010  *SHUUSEI
4020 '--------------------------------------------
4030 GET #1,SEC
4040 LOCATE 20,13 : PRINT SPC(40)
4050 COLOR 4 : LOCATE 28,13
4060 PRINT "  ｼｭｳｾｲ ﾅｹﾚﾊﾞ RET ｷｰ   "
4070 COLOR 0
4080 LOCATE 20,3 : PRINT "ﾅﾏｴ      :";NM$
4090 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A1$
4100 IF A1$="" THEN A1$=NM$
4110 PRINT TAB(20);"ﾕｳﾋﾞﾝ No:";PN$
```

```
4120 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A2$
4130 IF A2$="" THEN A2$=PN$
4140 PRINT TAB(20);"ｼﾞｳｼｮ     ";AD$
4150 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A3$
4160 IF A3$="" THEN A3$=AD$
4170 PRINT TAB(20);"ﾊﾞﾝﾁ      :";BA$
4180 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A4$
4190 IF A4$="" THEN A4$=BA$
4200 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
4210 COLOR 6 : LOCATE 28,13
4220 PRINT "  ﾄｳﾛｸ ｼﾏｽｶ ( y/n ) ?  "
4230 COLOR 0
4240 I$=INKEY$ : IF I$="" THEN GOTO 4240
4250 IF I$="Y" OR I$="y" THEN GOTO *KAKU
4260 IF I$="N" OR I$="n" THEN GOTO *KESU
4270 BEEP : BEEP : GOTO 4240
4280 '
4290 *KAKU
4300 LSET NM$=A1$ : LSET PN$=A2$
4310 LSET AD$=A3$ : LSET BA$=A4$
4320 PUT #1,SEC
4330 KAERU=-1
4340 '
4350 *KESU
4360 LOCATE 0,3
4370 FOR I=1 TO 8
4380   PRINT TAB(20);SPC(40)
4390 NEXT I
4400 LOCATE 20,13 : PRINT SPC(40)
4410 '
4420 IF KAERU THEN KAERU=0 : RETURN
4430 GOTO 4040
```

4・7 宛て名の印刷

さて、いよいよ最後の印刷プログラムです。

ここで扱うプログラムは、ディスクに登録してあるデータすべての宛て名を印刷するのではなく、ダイレクトメールなどの発送の必要のある人だけをプリンタに出力することにします。

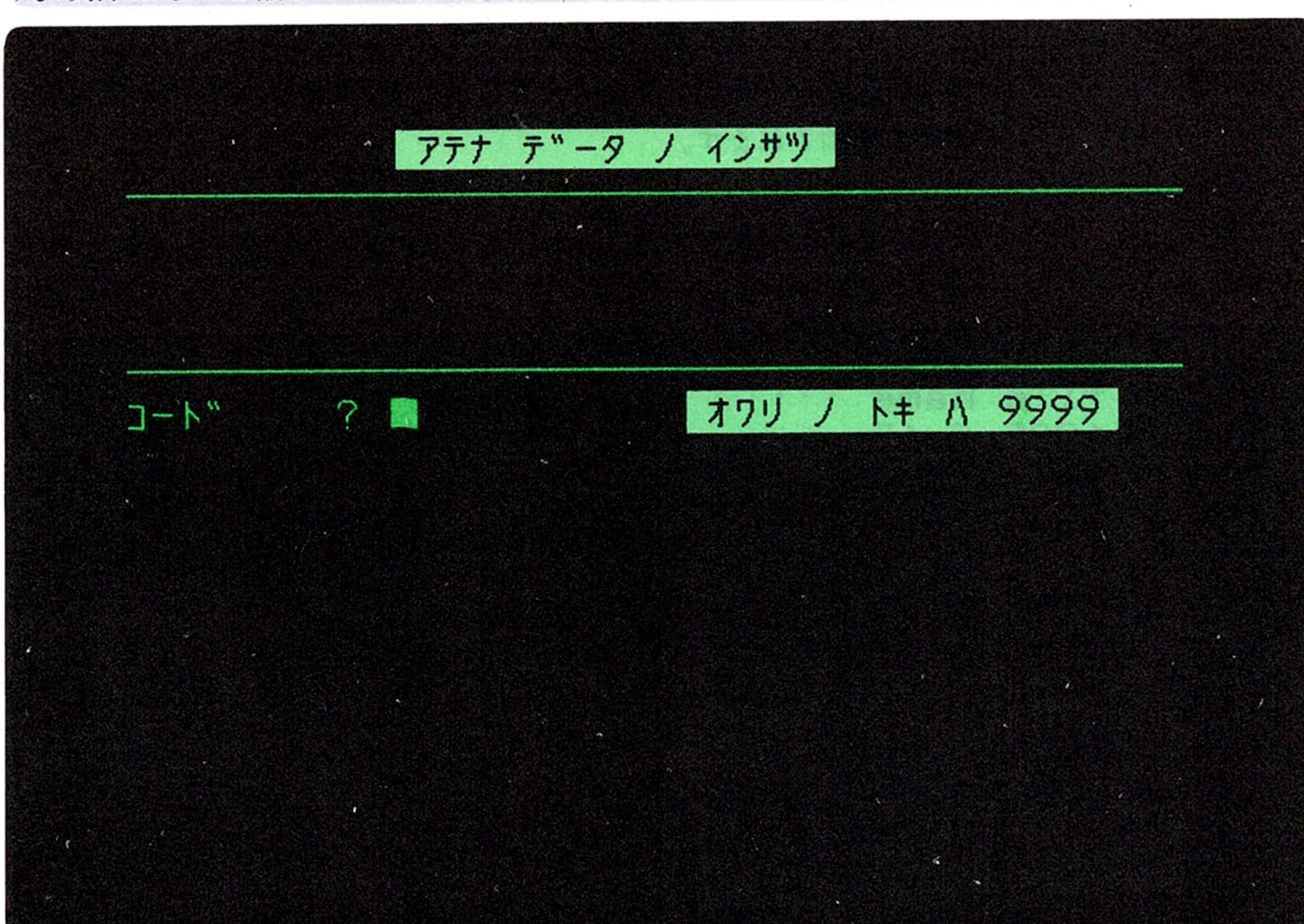
考え方は、対象となる人のコード番号をキーボードから入力し、それを一時、配列に退避させておいて、すべてのキー入力がすんだら、一括して印刷するシステムにします。

1件入力しては印刷する、というようにしてもかまわないのですが、大変時間の浪費になります。キー入力は人間がやらなければなりませんが、それさえすませたら、あとはプリンタにまかせましょう。

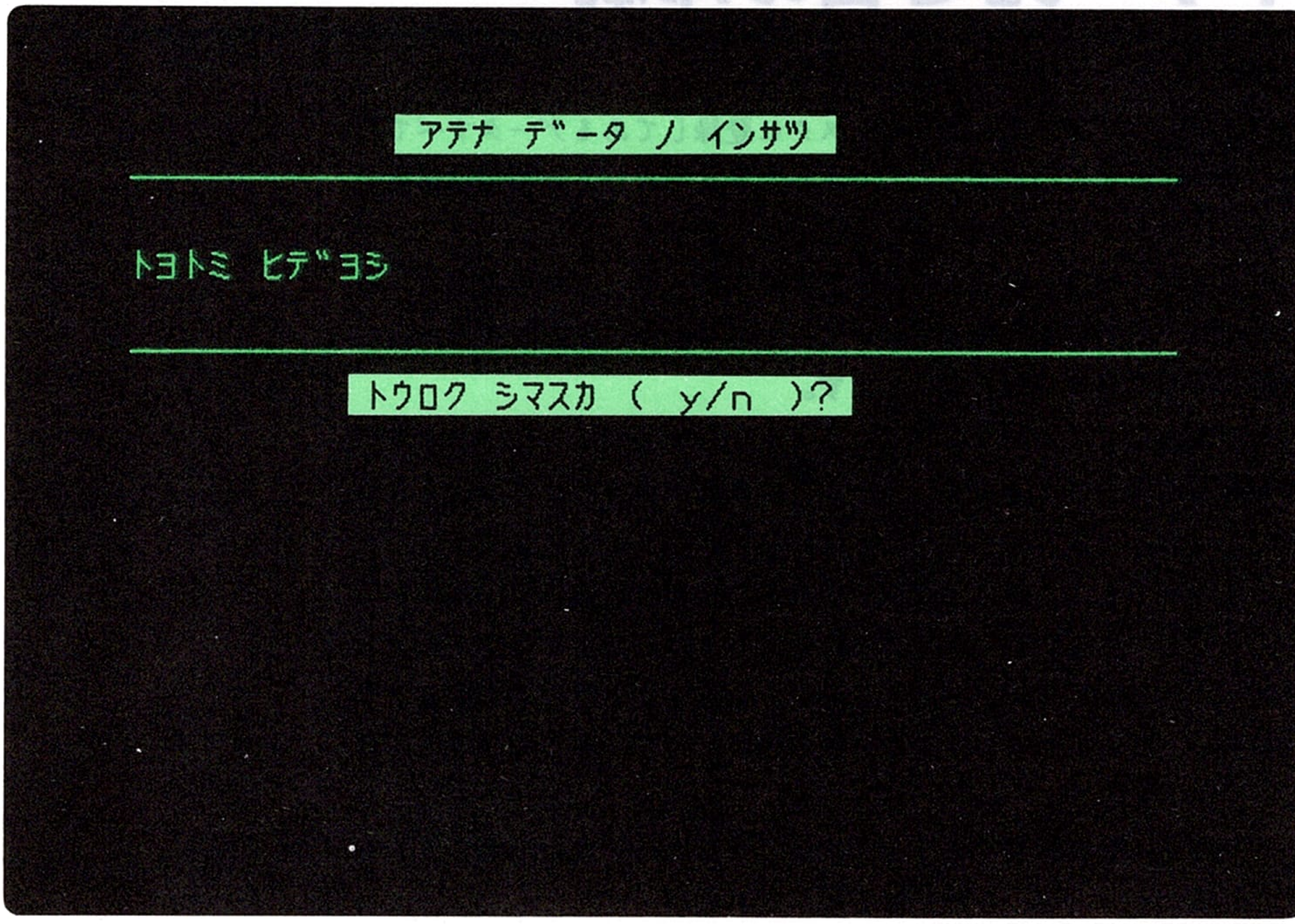
とはいっても、これは発送数が相当多数にわたる場合で、本節では2、3のデータをプリンタに打ち出してみるだけですから、プリント時間もつかの間ですが、一応発送数がふえることを想定して考えます。

しかし、データが多いか少ないかの相異だけで、基本的な考え方はまったく同じです。

プログラムを実行すると、まず次のようなコード入力画面になります。実際はもっと縮小された形で表示されます。



印刷したい人のコードを入力すると、登録してあるかどうかチェックし、もし登録してあれば次のような**確認画面**になります。



上のように氏名が表示されますから、YかNで答えてください。すると再びコード入力です。コードに対して9999が入力されれば宛て名印刷が開始されます。

さてプログラムを入力します。まず、ディスクのデータを配列にセットするまでです。

```
100 ' file name ····> atena3.bas
110 ' 1985.2.15
120 ' ｱﾃﾅ ﾉ ｲﾝｻﾂ
130 ' save "atena3.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C),A(C) : C=0 : GOSUB *YOMU
210 '
```

配列CDはコードを格納しておく配列ですが、目的のデータが見つかった場合に、そのデータが存在するセクタ番号を退避させておく**配列A**をあらたに宣言します。印刷のときは、この配列の内容に相当するセクタを順次バッファ上に取り出し、LPRINT文でプリンタへ出力します。

ファイルオープン、インデックスの読み出し部分は前節と同じです。

```
1000 '────────────────────────────
1010  *FOPEN
1020 '────────────────────────────
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
2000 '────────────────────────────
2010  *YOMU
2020 '────────────────────────────
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
```

次は、入力画面の作成です。

これも前節と同じですが、LOCATE文のY座標位置が違います。

```
220 LOCATE 28,0 : COLOR 4
230 PRINT "   宛て名データの印刷  " : COLOR 0
240 LOCATE 20,1 : PRINT STRING$(40,149)
250 LOCATE 20,5 : PRINT STRING$(40,149)
260 '
```

コード入力部分

次のプログラムを入力してください。

```
270 '───────────────────────────────
280 *LOOP
290 '───────────────────────────────
300 LOCATE 40,6 : COLOR 6
310 PRINT "  オワリ ノ トキ ハ 9999 " : COLOR 0
320 LOCATE 20,6 : INPUT "コード    ";CD
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)  ←該当コードがあるかどうか探す
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI  ←コードが見つからないとき
360 SEC=BANGOO
370 GOSUB *KAKUNIN  ←確認サブルーチンへ
380 IF NOT TOOROKU THEN GOTO 420  ←キャンセル
390 T=T+1 : TOOROKU=0
400 IF T>C THEN T=T-1 : GOTO *OWARI
410 A(T)=SEC  ←セクタ番号を配列に格納する
420 GOTO *LOOP
430 '
```

コメントを入れておきましたので理解できると思います。

コード入力があったら**SEARCH関数**で配列CDの中を探し、もし見つからなければ次のサブルーチンへ飛ばします。

```
3000 '───────────────────────────────
3010  *MITSUKARANAI
3020 '───────────────────────────────
3030 LOCATE 20,6 : PRINT SPC(40)
3040 LOCATE 30,6 : COLOR 6
3050 PRINT " トウロク シテ アリマセン !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,6 : PRINT SPC(40)
3090 RETURN *LOOP
3100 '
```

この場合、「トウロク　シテ　アリマセン」というエラーメッセージを表示して、再びコード入力へ戻ります。**RETURN文で戻り先の指定をしている**ことに注意してください。

該当コードがあった場合には、次の**確認サブルーチン**へジャンプします。

```
4000 '
4010  *KAKUNIN
4020 '
4030 GET #1,SEC
4040 LOCATE 20,3
4050 PRINT NM$
4060 LOCATE 20,6 : PRINT SPC(40)
4070 LOCATE 28,6 : COLOR 6
4080 PRINT " ﾄｳﾛｸ ｼﾏｽｶ ( y/n )? "
4090 COLOR 0
4100 I$=INKEY$ : IF I$="" THEN GOTO 4100
4110 IF I$="Y" OR I$="y" THEN GOTO 4140
4120 IF I$="N" OR I$="n" THEN GOTO 4150
4130 BEEP : GOTO 4100
4140 TOOROKU=-1
4150 LOCATE 20,3 : PRINT SPC(40)
4160 LOCATE 20,6 : PRINT SPC(40)
4170 RETURN
4180 '
```

キャンセルした場合には、**変数TOOROKUの内容は０**(「**偽**」であるということ)のままサブルーチンから帰りますから、380行の条件文にひっかかり420行へ飛ばされます。

登録する場合にはカウンタＴを１つ繰り上げ、これを添字として、410行で**配列Ａにセクタ番号を格納**しておきます。

400行の条件文は、印刷しようとするデータ数がディスクに実際に登録されているデータ数より多くなった場合、410行で、

Subscript out of range

のエラーになってしまいますので、このようなことはほとんどないと思いますが、一応いれておきました。

データがディスクには10個しか書き込まれていないのに、勘違いして１つのデータを重複して登録してしまったときなどがこのケースに相当します。

390行で変数TOOROKUの内容をゼロクリアしているように、一度**その使命を果たしたらリセットしておくこと**を忘れないようにしてください。これを忘れると、キャンセルしてもデータはプリンタへ出力

されてしまいます。

宛て名印刷サブルーチン

この部分は、終りの処理に含めることにしました。

```
440 '─────────────────────
450  *OWARI
460 '─────────────────────
470 CLS
480 LOCATE 30,3 : COLOR 4
490 PRINT "    ｲﾝｻﾂ ﾁｭｳ    " : COLOR 0
500 GOSUB *INSATSU
510 LOCATE 30,3
520 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
530 FOR I=1 TO 2000 : NEXT I
540 RUN "menu03"
550 '
```

「インサツ　チュウ」を表示してから、次の印刷サブルーチンへジャンプさせます。

```
5000 '─────────────────────
5010  *INSATSU
5020 '─────────────────────
5030 FOR I=1 TO T
5040   GET #1,A(I)  ←────── バッファにデータを取り出す
5050   LPRINT PN$                                  ┐
5060   LPRINT AD$                                  │
5070   LPRINT TAB(15);BA$                          ├─印刷
5080   LPRINT                                      │
5090   LPRINT TAB(10);NM$;TAB(30);" ｻﾏ"            ┘
5100   FOR J=1 TO 5 : LPRINT : NEXT J ←┐
5110 NEXT I                            │
5120 CLOSE : RETURN   前の宛て名との間に5行分空白をとる
```

配列Aの中に、印刷すべきデータが存在するディスク上のセクタ番号が格納されていますから、これを順次バッファ上にGET文を使って取り出し、これをプリンタへ出力します。

紙送り

5100行は、**任意の行数だけ紙送りをする常とう手段**です。

プログラムをセーブしてから実行してみてください。

次に示すのは、この章で扱ったデータをプリンタに出力したものです。あなたもためしてみてください。

また、いろいろ集計データや資料などを出力するルーチンを作り、宛て名書きをしたあとすぐに出力させるようにして、窓つき封筒を利用して発送すれば、かなり実用化できると思います。

```
111
aaaaaaaaaaaaaaaa
                         1-1-1

                  ﾄｸｶﾞﾜ ｲｴﾔｽ                    ｻﾏ


444
dddddddddddddddd
                         4-4-4

                  ｱｹﾁ ﾐﾂﾋﾃﾞ                     ｻﾏ


222
bbbbbbbbbbbbbbbb
                         2-2-2

                  ﾄﾖﾄﾐ ﾋﾃﾞﾖｼ                    ｻﾏ


333
cccccccccccccccc
                         3-3-3

                  ｵﾀﾞ ﾉﾌﾞﾅｶﾞ                    ｻﾏ
```

次に本節でのプログラムリストを示しておきます。

```
100 ' file name ････> atena3.bas
110 ' 1985.2.15
120 ' ｱﾃﾅ ﾉ ｲﾝｻﾂ
130 ' save "atena3.bas"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C),A(C) : C=0 : GOSUB *YOMU
210 '
220 LOCATE 28,0 : COLOR 4
230 PRINT "   ｱﾃﾅ ﾃﾞｰﾀ ﾉ ｲﾝｻﾂ   " : COLOR 0
240 LOCATE 20,1 : PRINT STRING$(40,149)
250 LOCATE 20,5 : PRINT STRING$(40,149)
260 '
270 '────────────────────────────────────────
280 *LOOP
290 '────────────────────────────────────────
300 LOCATE 40,6 : COLOR 6
310 PRINT "  ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999 " : COLOR 0
320 LOCATE 20,6 : INPUT "ｺｰﾄﾞ     ";CD
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI
360 SEC=BANGOO
370 GOSUB *KAKUNIN
380 IF NOT TOOROKU THEN GOTO 420
390 T=T+1 : TOOROKU=0
400 IF T>C THEN GOTO *OWARI
410 A(T)=SEC
420 GOTO *LOOP
430 '
440 '────────────────────────────────────────
450  *OWARI
460 '────────────────────────────────────────
470 CLS
480 LOCATE 30,3 : COLOR 4
490 PRINT "   ｲﾝｻﾂ ﾁｭｳ    " : COLOR 0
500 GOSUB *INSATSU
510 LOCATE 30,3
520 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
530 FOR I=1 TO 2000 : NEXT I
540 RUN "menu03"
550 '
```

```
1000 '─────────────────────────────
1010  *FOPEN
1020 '─────────────────────────────
1030 OPEN "2:atena1.dat" AS #1
1040 FIELD #1,20 AS NM$,6 AS PN$,35 AS AD$,20 AS BA$
1050 RETURN
1060 '
2000 '─────────────────────────────
2010  *YOMU
2020 '─────────────────────────────
2030 OPEN "2:index1.dat" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
3000 '─────────────────────────────
3010  *MITSUKARANAI
3020 '─────────────────────────────
3030 LOCATE 20,6 : PRINT SPC(40)
3040 LOCATE 30,6 : COLOR 6
3050 PRINT " ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,6 : PRINT SPC(40)
3090 RETURN *LOOP
3100 '
4000 '─────────────────────────────
4010  *KAKUNIN
4020 '─────────────────────────────
4030 GET #1,SEC
4040 LOCATE 20,3
4050 PRINT NM$
4060 LOCATE 20,6 : PRINT SPC(40)
4070 LOCATE 28,6 : COLOR 6
4080 PRINT " ﾄｳﾛｸ ｼﾏｽｶ ( y/n )?  "
4090 COLOR 0
4100 I$=INKEY$ : IF I$="" THEN GOTO 4100
4110 IF I$="Y" OR I$="y" THEN GOTO 4140
4120 IF I$="N" OR I$="n" THEN GOTO 4150
4130 BEEP : GOTO 4100
4140 TOOROKU=-1
4150 LOCATE 20,3 : PRINT SPC(40)
4160 LOCATE 20,6 : PRINT SPC(40)
4170 RETURN
```

```
4180 '
5000 '─────────────────────────────────────
5010  *INSATSU
5020 '─────────────────────────────────────
5030 FOR I=1 TO T
5040   GET #1,A(I)
5050   LPRINT PN$
5060   LPRINT AD$
5070   LPRINT TAB(15);BA$
5080   LPRINT
5090   LPRINT TAB(10);NM$;TAB(30);" ｻﾏ"
5100   FOR J=1 TO 5 : LPRINT : NEXT J
5110 NEXT I
5120 RETURN
```

第 5 章

ランダムファイル[II]

5・1　ディスクの有効利用

第4章の宛て名書きプログラム、いかがでしたか？　ディスクとのデータのやりとり、ご理解いただけたでしょうか。

ランダムデータファイル管理の基本は、おおよそ第4章に述べたとおりです。数値データを扱いませんでしたが、MKI$、MKS$、MKD$を使ってディスクに保存し、使いたいときにCVI、CVS、CVDで数値に変換する、ということさえわかってしまえば、あとは加減乗除をしてプリンタに出力するとか、要するにN_{88}-BASICでの操作と変わりありません。

ディスクを使ったらカセットテープはもう使う気にならない、とよく耳にしますが、それはおそらく、データを保存する容量がきわめて大きい、セーブ・ロードに時間がかからない、といったあたりに理由があるのではないでしょうか。

しかし、容量が大きいといっても、できるだけ有効に利用するにこしたことはありません。私たちが学んだ、前章でのデータファイルの使い方はどうだったでしょうか。**氏名、郵便番号、住所、番地で81セクタを使い、1セクタの残り175バイトは未使用でした。**ずいぶん**ムダな使い方**をしていたわけです。

この章では、ディスクの有効利用をテーマに話を進めますが、できたら話につながりがあったほうがよいと思いますので、前章のプログラムを例にとって、一部を変更し、一部ステートメントを追加し、ご説明したいと思います。

ムダのないデータ設計

前章では1セクタ内に1人分のデータを書き込んでいました。

今度は下の図のように3人分のデータを保存することにしましょう。**1名で85バイト使いますから、3名で255バイト、つまり1バイトだけ未使用ということになります。**ほぼ**100パーセント活用**です。

〈注〉前章では住所に35バイト使っていましたが本章では39バイトに変更します。

氏　名(20バイト)	郵便番号(6バイト)	住　所(39バイト)	番　地(20バイト)
氏　名(20バイト)	郵便番号(6バイト)	住　所(39バイト)	番　地(20バイト)
氏　名(20バイト)	郵便番号(6バイト)	住　所(39バイト)	番　地(20バイト)

←――― 1バイト未使用

さて、１セクタ内の**データフォーマット**が決まりましたから、まず問題になるのは**フィールドの割り当て**です。前章で、

データフォーマット

フィールド割り当て

```
FIELD #1,20 AS NM$,6 AS PN$,39 AS AD$,20 AS BA$
```

のように割り当てていましたので、同じ考え方をして次のように定義すればよいのでしょうか？

```
FIELD #1,20 AS N1$,6 AS P1$,39 AS A1$,20 AS B1$
FIELD #1,20 AS N2$,6 AS P2$,39 AS A2$,20 AS B2$
FIELD #1,20 AS N3$,6 AS P3$,39 AS A3$,20 AS B3$
```

一見だまされてしまいそうな感じですが、この場合、３人分の各項目が同じFIELDを共有していることになり、これでPUT、GETをすると、データが混同してしまいます。

ですから、バッファ上に３人分のフィールドを確保したいのでしたら、すべてコンマで区切って次のようにつなげて定義すればよいのです。

Ⓐ

```
FIELD #1,20 AS N1$,6 AS P1$,39 AS A1$,20 AS B1$
        ,20 AS N2$,6 AS P2$,39 AS A2$,20 AS B2$
        ,20 AS N3$,6 AS P3$,39 AS A3$,20 AS B3$
```

そしてこれを少し変形して次のようにしても、それぞれの氏名、郵便番号、住所、番地に応じたフィールド幅を確保することができます。

Ⓑ

```
❶ FIELD #1,  0 AS D$,20 AS N1$,6 AS P1$,39 AS A1$,20 AS B1$
❷ FIELD #1, 85 AS D$,20 AS N2$,6 AS P2$,39 AS A2$,20 AS B2$
❸ FIELD #1,170 AS D$,20 AS N3$,6 AS P3$,39 AS A3$,20 AS B3$
```

これはどういうことかと申しますと、まず❶のFIELD文で、**先頭のD＄は０ですから、先頭にＮ１＄が入り順次Ｂ１＄まで設定されます。**次に❷であらためてFIELD文が定義され、まず**85バイトのＤ＄が入り、そのうしろに続いてＮ２＄～Ｂ２＄が設定されます。**

ここに登場する**Ｄ＄はダミー**で、実際には何も利用されないものです。そしてこの85バイトという数は、❶のFIELD文で定義されている変数のフィールド幅を合計したものです。

ダミー変数を使う

つまり、❶でＮ１＄～Ｂ１＄のフィールド域が確保され、❷ではそのフィールド幅をもったダミーを踏み台にしてＮ２＄～Ｂ２＄のフィールド域が決められています。ということはバッファ上ではＮ１＄からＢ２＄までのフィールド域が作られたということになります。

同じように、❸のFIELD文によって、N１＄からＢ３＄までが確保されたことになります。

❶～❸の過程で、**ダミーのＤ＄は０、85、170バイトと変化しますが、最終的には170バイトが残ります。**

しかしこのＤ＄、まったくのダミーですから、プログラムでGET、PUTさえしなければよいのです。

さて、これでだいぶFIELD文の下準備ができてきました。

要は、ダミー変数を使えば、そのうしろに希望のフィールド域を確保できるということを理解していただければよいと思います。

ここでもう一度前に戻ってⒶのFIELD文を次のように配列を使って書き換えてみましょう。

```
FIELD #1,20 AS N$(1),6 AS P$(1),39 AS A$(1),20 AS B$(1)
        ,20 AS N$(2),6 AS P$(2),39 AS A$(2),20 AS B$(2)
        ,20 AS N$(3),6 AS P$(3),39 AS A$(3),20 AS B$(3)
```

そしてⒷも書き換えてみます。

```
FIELD #1,  0 AS D$,20 AS N$(1),6 AS P$(1),39 AS A$(1),20 AS B$(1)
FIELD #1, 85 AS D$,20 AS N$(2),6 AS P$(2),39 AS A$(2),20 AS B$(2)
FIELD #1,170 AS D$,20 AS N$(3),6 AS P$(3),39 AS A$(3),20 AS B$(3)
```

もうおわかりと思います。

いくらなんでも同じ形のFIELD文を３行も長々と書くのはちょっとナンセンスですね。FOR～NEXT文でもう少しすっきりさせたいものです。

そうです、次のように表せますね。

```
200 FOR I=1 TO 3
210   FIELD #1,85*(I-1) AS D$,20 AS N$(I),6 AS P$(I)
                           ,39 AS A$(I),20 AS B$(I)
220 NEXT I
```

ずいぶんさっぱりした形に表現できました。

このように、バッファをいくつかのレコードで使用する場合には、文字変数をたくさんつらねたりしないで、FOR～NEXT文によって処理できるのです。それに、ダミーの考え方、これも大切ですからぜひ体得してください。

さて、前章では１セクタに１レコードを書き込んでいましたが、この例では１セクタに３レコードを書き込めます。すると、PUT、GETはどうしたらよいのでしょうか？

GET、PUTするセクタ番号、そのセクタ内で何番目のレコードかがわかればよいわけですから、**インデックスファイルのレコード番号を3で割った商、余りを利用する**ことによって求められます。この辺は具体例で学ぶことにしましょう。

これから、前章で作ったプログラムを一部変更・追加して、1セクタに3名分書き込めるようにしようというわけですが、**一番大きな変更点はFIELD文です。**

そして、いちばん**大切な追加点はセクタ番号、セクタ内でのレコード番号を計算するステートメントです。**

すでにでき上がっているプログラムに手を加えてということですから、条件文など、多少まわりくどい点もやむをえないと思います。

しかし、個人で、自分の仕事に合ったプログラムを作ろうとするとき、やはり最初に小さなプログラムを作って、次に、あんなこともしたい、こんなこともしたいと欲がでてきて、変更・追加をくりかえして、プログラムをふくらませていくのです。

それでは、プログラムの手直しにかかりますが、順を追っての説明はやめ、修正後の全プログラムリストをかかげ、**追加点は緑色の文字で、修正個所は緑色のアミで**示すことにしました。

〈注〉本節でのフィールドの定義のしかたは、重ねて定義するところから「フィールドの多重定義」といいます。ここでは紙数の関係から詳しく説明できませんでしたが、本書の続編で具体的にわかりやすく解説します。

5・2　データ書き込みプログラム

まず、183ページのプログラムをロードしてください。

```
load "atena1.bas" ⏎
```

ロードできましたら、次のプログラムリストを見ながら追加行をキーインし、修正箇所はなおしてください。

```
100 '  file name ····> ATENA1.BAS
110 '  1985.2.15
120 '  ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ﾍﾉ ｶｷｺﾐ
130 '  save "ATENA1.BAS"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
165 DIM NM$(3),PN$(3),AD$(3),BA$(3)
170 '
180 GOSUB *FOPEN
190 IF LOF(1)<>0 THEN GOSUB *COUNT : GOTO 220
200 OPEN "2:INDEX1.DAT" FOR OUTPUT AS #2
210 RN=1 : GOTO *NYUURYOKU
220 OPEN "2:INDEX1.DAT" FOR APPEND AS #2
230 '
240 '----------------------------------------------------------
250  *NYUURYOKU
260 '----------------------------------------------------------
270 LOCATE 28,0 : COLOR 4
280 PRINT "      ｱﾃﾅ ﾉ ﾆｭｳﾘｮｸ        " : COLOR 0
290 LOCATE 20,1 : PRINT STRING$(40,149)
300 LOCATE 20,9 : PRINT STRING$(40,149)
310 '
320 *LOOP
330 LOCATE 40,3 : COLOR 6
340 PRINT " ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999  " : COLOR 0
350 LOCATE 20,3 : INPUT "ｺｰﾄﾞ      ";CD
360 IF CD=9999 THEN GOTO *OWARI
370 LOCATE 40,3 : PRINT SPC(20)
380 LOCATE 20,4 : INPUT "ﾅﾏｴ       ";A1$
390 LOCATE 20,5 : INPUT "ﾕｳﾋﾞﾝ No";A2$
```

```
400 LOCATE 20,6 : INPUT "ｼﾞｭｳｼｮ   ";A3$
410 LOCATE 20,7 : INPUT "ﾊﾞﾝﾁ     ";A4$
420 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
430 LOCATE 28,10 : COLOR 6
440 PRINT "  ﾄｳﾛｸ ｼﾏｽｶ ( y/n )? "
450 COLOR 0
460 I$=INKEY$ : IF I$="" THEN GOTO 460
470 IF I$="Y" OR I$="y" THEN GOTO *KAKU
480 IF I$="N" OR I$="n" THEN GOTO *KESU
490 BEEP : BEEP : GOTO 460
500 '
510 *KAKU
512 SEC=(RN-1) ¥ 3+1
514 REC=(RN-1) MOD 3+1
516 IF REC>1 THEN GET #1,SEC
520 WRITE #2,CD
530 LSET NM$(REC)=A1$
540 LSET PN$(REC)=A2$
550 LSET AD$(REC)=A3$
560 LSET BA$(REC)=A4$
570 PUT #1,SEC
580 RN=RN+1
590 '
600 *KESU
610 LOCATE 0,3
620 FOR I=1 TO 5
630   PRINT TAB(20);SPC(40)
640 NEXT I
650 LOCATE 20,10 : PRINT SPC(40)
660 GOTO *LOOP
670 '
680 '────────────────────────────────
690  *OWARI
700 '────────────────────────────────
710 CLOSE : CLS
720 LOCATE 30,3
730 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
740 FOR I=1 TO 5000 : NEXT I
750 RUN "menu04"
760 '
1000 '────────────────────────────────
1010  *FOPEN
```

```
1020 '─────────────────────────────
1030 OPEN "2:ATENA1.DAT" AS #1
1035 FOR I=1 TO 3
1040 FIELD #1,85*(I-1) AS D$,20 AS NM$(I),
       6 AS PN$(I),39 AS AD$(I),20 AS BA$(I)
1045 NEXT I
1050 RETURN
1060 '
2000 '─────────────────────────────
2010  *COUNT
2020 '─────────────────────────────
2030 OPEN "2:INDEX1.DAT" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   INPUT #2,DUMMY
2060   RN=RN+1
2070 WEND
2080 RN=RN+1
2090 CLOSE #2
2100 RETURN
```

プログラムファイル名、データファイル名はいずれも大文字にして前章のファイル名と区別するようにしました。

165行の**DIM文**は、FIELD文の定義に使う変数のための配列ですが、これはなくてもかまいません。配列を宣言しなければ、自動的に10個分は確保されますが、このように、**使用する大きさだけ宣言しておけば、メモリの節約になる**と考えたからです。

メモリの節約

2060行のRNはレコードナンバーの頭文字をとったもので、この変数には登録順位が入れられます。

そして、このRNをもとにして**書き込むセクタ番号**、および**そのセクタ内でのレコード番号**（1つのセクタに3レコード書き込まれますが、何番目のレコードかということ）を算出します。

セクタ番号の計算

RNの内容から1を引き、それを3で割った商に1を加えた式

(RN−1)¥3+1

が、求めるセクタ番号になります。¥は商を求める演算記号で

S=5¥3

とすると、商が1、余りが2ですからSには1が代入されます。

レコード番号の計算

次に、**セクタ内でのレコード番号**を求めるには、整数除算の余りを

求めるMODを使い、

(RN−1) MOD 3+1

とします。RNから1を引き、その値を3で割って、その余りに1を加えます。ですから

R=5 MOD 3+1

ですと、Rには5を3で割った余り2に1を足した3が入ります。

これらの計算を512行、514行で行っています。

516のGET文はなぜ入れておくかといいますと、570行で1セクタ分まるまるPUTするわけですから、セクタ内の2番目以降のレコードの場合には、ここでGETしてすでにファイルに書き込まれている同じセクタ内のデータをバッファ上に取り出しませんと、570行のPUT文で消えてしまいます。

たとえば、あるセクタの第1、第2レコードはすでに書き込まれていたとします。そして今回3番目のレコードを入力しました。このとき、このセクタをGETして第1、第2レコードもバッファ上に取り出しませんと、バッファ上の第1、第2レコードに割り当てられた変数には何も入っていませんから、この状態でPUTすると、第3レコードだけが書き込まれ、第1、第2レコードはクリアされてしまいます。このミスはよくありますので注意してください。

580行ではレコード番号を1進めています。

1035行から1045行でフィールドを定義しています。

次にデータ修正のプログラム、印刷のプログラム、メニュープログラムと順に直していきますが、変更・追加個所は、いま説明したようなところです。プログラムを追ってみてください。

プログラムの追加・修正が終ったら必ずこのプログラムを新しいファイル名でセーブしておきましょう。

5·3 データ修正プログラム

```
100 ' file name ････> ATENA2.BAS
110 ' 1985.2.15
120 ' ﾗﾝﾀﾞﾑ ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ
130 ' save "ATENA2.BAS"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
165 DIM NM$(3),PN$(3),AD$(3),BA$(3)
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C) : C=0 : GOSUB *YOMU
210 '
220 LOCATE 28,0 : COLOR 4
230 PRINT "  ｱﾃﾅ ﾃﾞｰﾀ ﾉ ｼｭｳｾｲ   " : COLOR 0
240 LOCATE 20,1  : PRINT STRING$(40,149)
250 LOCATE 20,12 : PRINT STRING$(40,149)
260 '
270 '-------------------------------------------
280 *LOOP
290 '-------------------------------------------
300 LOCATE 40,13 : COLOR 6
310 PRINT "  ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999 " : COLOR 0
320 LOCATE 20,13 : INPUT "ｺｰﾄﾞ    ";CD
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI
360 SEC=(BANGOO-1) ¥ 3+1
365 REC=(BANGOO-1) MOD 3+1
370 GOSUB *SHUUSEI
380 GOTO *LOOP
390 '
400 '-------------------------------------------
410  *OWARI
420 '-------------------------------------------
430 CLOSE  : CLS
440 LOCATE 25,3
450 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
```

```
460 FOR I=1 TO 2000 : NEXT I
470 RUN "menu04"
480 '
1000 '───────────────────────────
1010  *FOPEN
1020 '───────────────────────────
1030 OPEN "2:ATENA1.DAT" AS #1
1035 FOR I=1 TO 3
1040 FIELD #1,85*(I-1) AS D$,20 AS NM$(I),
       6 AS PN$(I),39 AS AD$(I),20 AS BA$(I)
1045 NEXT I
1050 RETURN
1060 '
2000 '───────────────────────────
2010  *YOMU
2020 '───────────────────────────
2030 OPEN "2:INDEX1.DAT" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
3000 '───────────────────────────
3010  *MITSUKARANAI
3020 '───────────────────────────
3030 LOCATE 20,13 : PRINT SPC(40)
3040 LOCATE 30,13 : COLOR 6
3050 PRINT " ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,13 : PRINT SPC(40)
3090 RETURN *LOOP
3100 '
4000 '───────────────────────────
4010  *SHUUSEI
4020 '───────────────────────────
4030 GET #1,SEC
4040 LOCATE 20,13 : PRINT SPC(40)
```

```
4050 COLOR 4 : LOCATE 28,13
4060 PRINT "  ｼｭｳｾｲ ﾅｹﾚﾊﾞ RET ｷｰ   "
4070 COLOR 0
4080 LOCATE 20,3 : PRINT "ﾅﾏｴ      :";NM$(REC)
4090 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A1$
4100 IF A1$="" THEN A1$=NM$(REC)
4110 PRINT TAB(20);"ﾕｳﾋﾞﾝ No:";PN$(REC)
4120 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A2$
4130 IF A2$="" THEN A2$=PN$(REC)
4140 PRINT TAB(20);"ｼﾞｭｳｼｮ   ";AD$(REC)
4150 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A3$
4160 IF A3$="" THEN A3$=AD$(REC)
4170 PRINT TAB(20);"ﾊﾞﾝﾁ      :";BA$(REC)
4180 PRINT TAB(22);"ｼｭｳｾｲ"; : INPUT A4$
4190 IF A4$="" THEN A4$=BA$(REC)
4200 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
4210 COLOR 6 : LOCATE 28,13
4220 PRINT "  ﾄｳﾛｸ ｼﾏｽｶ ( y/n ) ?  "
4230 COLOR 0
4240 I$=INKEY$ : IF I$="" THEN GOTO 4240
4250 IF I$="Y" OR I$="y" THEN GOTO *KAKU
4260 IF I$="N" OR I$="n" THEN GOTO *KESU
4270 BEEP : BEEP : GOTO 4240
4280 '
4290 *KAKU
4300 LSET NM$(REC)=A1$ : LSET PN$(REC)=A2$
4310 LSET AD$(REC)=A3$ : LSET BA$(REC)=A4$
4320 PUT #1,SEC
4330 KAERU=-1
4340 '
4350 *KESU
4360 LOCATE 0,3
4370 FOR I=1 TO 8
4380   PRINT TAB(20);SPC(40)
4390 NEXT I
4400 LOCATE 20,13 : PRINT SPC(40)
4410 '
4420 IF KAERU THEN KAERU=0 : RETURN
4430 GOTO 4040
```

5·4 宛て名印刷プログラム

```
100 ' file name ····> ATENA3.BAS
110 ' 1985.2.15
120 ' ｱﾃﾅ ﾉ ｲﾝｻﾂ
130 ' save "ATENA3.BAS"
140 '
150 WIDTH 80,20 : CONSOLE 0,20,0,0
160 DEFINT A-Z
165 DIM NM$(3),PN$(3),AD$(3),BA$(3)
170 '
180 GOSUB *FOPEN
190 GOSUB *YOMU : NIKAIME=-1
200 DIM CD(C),A(C),B(C) : C=0 : GOSUB *YOMU
210 '
220 LOCATE 28,0 : COLOR 4
230 PRINT "   ｱﾃﾅ ﾃﾞｰﾀ ﾉ ｲﾝｻﾂ   " : COLOR 0
240 LOCATE 20,1 : PRINT STRING$(40,149)
250 LOCATE 20,5 : PRINT STRING$(40,149)
260 '
270 '―――――――――――――――――――――――――――――――――――
280 *LOOP
290 '―――――――――――――――――――――――――――――――――――
300 LOCATE 40,6 : COLOR 6
310 PRINT "  ｵﾜﾘ ﾉ ﾄｷ ﾊ 9999 " : COLOR 0
320 LOCATE 20,6 : INPUT "ｺｰﾄﾞ    ";CD
330 IF CD=9999 THEN GOTO *OWARI
340 BANGOO=SEARCH(CD,CD)
350 IF BANGOO=-1 THEN GOSUB *MITSUKARANAI
360 SEC=(BANGOO-1) ¥ 3+1
365 REC=(BANGOO-1) MOD 3+1
370 GOSUB *KAKUNIN
380 IF NOT TOOROKU THEN GOTO 420
390 T=T+1 : TOOROKU=0
400 IF T>C THEN GOTO *OWARI
410 A(T)=SEC : B(T)=REC
420 GOTO *LOOP
430 '
440 '―――――――――――――――――――――――――――――――――――
450  *OWARI
```

```
460 '──────────────────────────────
470 CLS
480 LOCATE 30,3 : COLOR 4
490 PRINT "   ｲﾝｻﾂ ﾁｭｳ   " : COLOR 0
500 GOSUB *INSATSU
510 LOCATE 30,3
520 PRINT "ﾒﾆｭｰ ﾍ ﾓﾄﾞﾘﾏｽ"
530 FOR I=1 TO 2000 : NEXT I
540 RUN "menu04"
550 '
1000 '──────────────────────────────
1010  *FOPEN
1020 '──────────────────────────────
1030 OPEN "2:ATENA1.DAT" AS #1
1035 FOR I=1 TO 3
1040 FIELD #1,85*(I-1) AS D$,20 AS NM$(I),
       6 AS PN$(I),39 AS AD$(I),20 AS BA$(I)
1045 NEXT I
1050 RETURN
1060 '
2000 '──────────────────────────────
2010  *YOMU
2020 '──────────────────────────────
2030 OPEN "2:INDEX1.DAT" FOR INPUT AS #2
2040 WHILE NOT EOF(2)
2050   C=C+1
2060   IF NIKAIME THEN GOTO 2080
2070   INPUT #2,DUMMY : GOTO 2090
2080   INPUT #2,CD(C)
2090 WEND
2100 CLOSE #2
2110 RETURN
2120 '
3000 '──────────────────────────────
3010  *MITSUKARANAI
3020 '──────────────────────────────
3030 LOCATE 20,6 : PRINT SPC(40)
3040 LOCATE 30,6 : COLOR 6
3050 PRINT " ﾄｳﾛｸ ｼﾃ ｱﾘﾏｾﾝ !! "
3060 BEEP : BEEP : COLOR 0
3070 FOR I=1 TO 2000 : NEXT I
3080 LOCATE 20,6 : PRINT SPC(40)
3090 RETURN *LOOP
```

```
3100 '
4000 '───────────────────────────────────
4010  *KAKUNIN
4020 '───────────────────────────────────
4030 GET #1,SEC
4040 LOCATE 20,3
4050 PRINT NM$(REC)
4060 LOCATE 20,6 : PRINT SPC(40)
4070 LOCATE 28,6 : COLOR 6
4080 PRINT " ﾄｳﾛｸ ｼﾏｽｶ ( y/n )?  "
4090 COLOR 0
4100 I$=INKEY$ : IF I$="" THEN GOTO 4100
4110 IF I$="Y" OR I$="y" THEN GOTO 4140
4120 IF I$="N" OR I$="n" THEN GOTO 4150
4130 BEEP : GOTO 4100
4140 TOOROKU=-1
4150 LOCATE 20,3 : PRINT SPC(40)
4160 LOCATE 20,6 : PRINT SPC(40)
4170 RETURN
4180 '
5000 '───────────────────────────────────
5010  *INSATSU
5020 '───────────────────────────────────
5030 FOR I=1 TO T
5035   SEC=A(I) : REC=B(I)
5040   GET #1,SEC
5050   LPRINT PN$(REC)
5060   LPRINT AD$(REC)
5070   LPRINT TAB(15);BA$(REC)
5080   LPRINT
5090   LPRINT TAB(10);NM$(REC);TAB(30);" ｻﾏ"
5100   FOR J=1 TO 5 : LPRINT : NEXT J
5110 NEXT I
5120 RETURN
```

5·5 メニュープログラム

```
100 '  file name ････> menu04
110 '  1985.2.15
120 '  ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ ﾌﾟﾛｸﾞﾗﾑ
130 '  save "menu04"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
230 '
240 PRINT TAB(20)
250 PRINT"┌──────────────────────────────┐"
260 PRINT TAB(20)
270 PRINT"│    ｱﾃﾅ ｲﾝｻﾂ ﾒﾆｭｰ (ｶｲﾃｲﾊﾞﾝ)    │"
280 PRINT TAB(20)
290 PRINT"└──────────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"-")
380 LOCATE 22,13
390 PRINT "ｼｺﾞﾄ ﾊﾞﾝｺﾞｳ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)
420 IF N>MENU OR N=0 THEN BEEP : GOTO 400
430 '
440 COLOR 4
450 LOCATE 22,N+3
```

```
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "    カクニン     "
510 COLOR 0 : LOCATE 26,15
520 PRINT "ジッコウ [,]  キャンセル [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
610 '─────────────────────────────────
620 ' menu data
630 '─────────────────────────────────
640 DATA 5
650 DATA データ ノ ニュウリョク ･････････････1,ATENA1.BAS
660 DATA データ ノ シュウセイ ･･･････････････2,ATENA2.BAS
670 DATA アテナ ノ インサツ ･････････････････3,ATENA3.BAS
680 DATA マスターメニュー ヘ モドル ･････････4,MASTER
690 DATA オワリ ･････････････････････････････5,dummy
```

第 6 章

マスターメニュープログラム

6・1　マスターメニュープログラム

これで本書も終りです。

簡単な英単語練習プログラムにはじまって宛て名印刷プログラムまで、シーケンシャルファイル、ランダムファイルを通して、データの扱い方について勉強してきました。

最初はちょっととっつきずらい面もあったかもしれませんが、ここまで進んでこられたみなさんは、ずいぶんなじんでいただいたのではないかと考えています。

さて、私たちがこれまでに作ってきた4つのプログラム

1　英単語練習プログラム
2　電話帳プログラム
3　宛て名印刷プログラム（No.1）
4　宛て名印刷プログラム（No.2）

には、それぞれメニュープログラムがついていて、メニューを選択することにより、データの修正・追加など、希望の処理をすることができました。

ここでは、さらに上の4つのプログラムを管理するマスターメニュープログラムを作っておくことにしましょう。

このマスターメニュープログラムをRUNさせると、いま述べた4つのプログラム名が表示され、希望のプログラムナンバーをキーインすると、さらにそのプログラムのメニューが表示されるというように、仕事が大変やりやすくなります。

つまり私たちがやるべきことは、システムを立ち上げて、このマスターメニューを走らせさえすれば、あとはコンピュータまかせということになります。

システムディスクの中のユーティリティプログラム dskut 2 .j88にはIDセクタを書き換えるためのメニューがあります。これを実行してIDセクタを書き換えれば、システム立ち上げから実行までを自動的にやってくれますが（これを**「オートスタート」**といいます)、ここでは説明を省略しますので、興味がおありの方は自分でやってみてください。

さあ、それではマスターメニュープログラムをキーインしてしまいましょう。

入力が終ったら、これをRUNさせて、いままでのプログラムが正常に走るかどうかテストしてみてください。

〈注〉このプログラムはわざわざ入力しなくとも、前節のプログラムをロードし、色アミの箇所を変更すれば完成します。

```
100 '  file name ････> MASTER
110 '  1985.2.15
120 ' マスター メニュー プ゜ログ゛ラム
130 '  save "MASTER"
140 '
150 WIDTH 80,25 : CONSOLE 0,25,0,0
160 DEFINT A-Z
170 '
180 READ MENU
190 DIM MENU$(MENU),PROG$(MENU)
200 FOR I=1 TO MENU
210   READ MENU$(I),PROG$(I)
220 NEXT I
230 '
240 PRINT TAB(20)
250 PRINT"┌──────────────────────────┐"
260 PRINT TAB(20)
270 PRINT"│        マスター メニュー        │"
280 PRINT TAB(20)
290 PRINT"└──────────────────────────┘"
300 PRINT
310 '
320 FOR I=1 TO MENU
330   LOCATE 22,I+3 : PRINT MENU$(I)
340 NEXT I
350 '
360 LOCATE 22,12
370 PRINT STRING$(28,"─")
380 LOCATE 22,13
390 PRINT "シコ゛ト ハ゛ンコ゛ウ ヲ ニュウリョク シテクタ゛サイ"
400 N$=INKEY$ : IF N$="" THEN GOTO 400
410 N=VAL(N$)
420 IF N>MENU OR N=0 THEN BEEP : GOTO 400
430 '
440 COLOR 4
450 LOCATE 22,N+3
```

```
460 PRINT MENU$(N)
470 COLOR 0
480 FOR I=1 TO 200 : BEEP 1 : BEEP 0 : NEXT I
490 LOCATE 30,14
500 COLOR 6 : PRINT "    ｶｸﾆﾝ     "
510 COLOR 0 : LOCATE 26,15
520 PRINT "ｼﾞｯｺｳ [,]  ｷｬﾝｾﾙ [.]"
530 I$=INKEY$ : IF I$="" THEN GOTO 530
540 IF I$="," AND N=MENU THEN CLS : END
550 IF I$="," THEN RUN PROG$(N)
560 IF I$<>"." THEN BEEP : GOTO 530
570 LOCATE 22,N+3 : PRINT MENU$(N)
580 LOCATE 22,14 : PRINT SPC(40)
590 LOCATE 22,15 : PRINT SPC(40)
600 GOTO 400
610 '―――――――――――――――――――――――――――――――――
620 '  menu data
630 '―――――――――――――――――――――――――――――――――
640 DATA 5
650 DATA ｴｲﾀﾝｺﾞ ﾉ ﾚﾝｼｭｳ ･･････････････1,menu01
660 DATA ﾃﾞﾝﾜﾁｮｳ ﾉ ｶﾝﾘ ･･････････････2,menu02
670 DATA ｱﾃﾅ ﾉ ｲﾝｻﾂ ･･････････････････3,menu03
680 DATA ｱﾃﾅ ﾉ ｲﾝｻﾂ (ｶｲﾃｲ ﾊﾞﾝ) ･････4,menu04
690 DATA ｵﾜﾘ ･･････････････････････････5,dummy
```

索引

あ行

か行

さ行

た行

な行

は行

ま行

や行

ら行

PC-8801mkIIFR/MR データファイル入門1

編　者　ナツメ出版企画　© Natsume shuppankikaku 1986
発行者　田村正隆
発行所　株式会社　ナツメ社
東京都千代田区神田神保町1－52(〒101)
電話　03(291)1257(代表)
振替　東京3－58661

印　刷　ラン印刷社
製　本　三修紙工

ISBN4-8163-0566-1

PC-8801MKⅡFR/MR
プログラミング＆オペレーション入門

ナツメ出版企画編―――1700円

●PC-8801MKⅡFR/MRは、従来のN88-BASIC V1はもとより、PC-8801MKⅡSRの持つアナログパレット機能、FM音源によるサウンド機能に加えて、さらに手軽に漢字が扱えるN88日本語BASICをも追加した多機能型パソコンで、ファミリーユースとしてもビジネスユースとしても、とにかく幅広く多方面に活用できます。本書では、さまざまな分野に題材を求め、プログラミングの基本から応用までをやさしく解説しました。この本を学べばプログラミングに相当自信がつくはずです。

【内容】◎電源を入れてみよう◎キー操作を覚えよう◎関数計算もできます◎画面をカラーに◎プログラムのセーブとロード◎表に数値を書き込む◎配列を使おう◎プログラムの完成◎ブロックくずし◎ボールが何に当たった？◎ボール表示とブロック消去◎算数教室◎プログラムのあらすじ◎たし算のサブルーチン◎もっとチャレンジしたければ◎DMプログラム◎データサーチ◎宛て名の印刷◎拡張命令について◎サウンド機能をいかそう◎タートルグラフィックの使い方◎グラフィック画面のモード◎円を描いて中を塗りつぶす◎ワールド座標に円を描く◎絵を描いてみよう◎ドレミファソラシド◎ボリュームとテンポ◎「禁じられた遊び」のテーマ音楽◎走れSL！◎プリンタの使い方◎日本語を入力してみよう◎プログラムの中で漢字をつかおう

PC-8801MKⅡ+SR+TR

実用版・ディスク実践プログラム集

安立隆一著―――1600円

●本書には、キーインするだけで即使える実用的なプログラム7本のリストとその使い方が掲載されています。扱ったプログラムは、「512色中36色イラスト作画プログラム」「成績管理プログラム」「成績順ソートプログラム」「スクリーン・ロードセーブ・プログラム」「顧客管理プログラム」「名前ソートプログラム」「スケジュール管理プログラム」です。

●本書に掲載したプログラムをすべて収録したディスケットサービスも行っています。媒体は5インチ両面倍密度ディスケット（5インチ2D）です。定価は8000円、送料350円です。ご注文の折は、「PC-8801MKⅡ＋SR＋TR 実用版・ディスク実践プログラム集専用ディスケット」と明記してください。

【内容】◎512色中36色イラスト作画プログラム◎成績管理プログラム◎使用システムと立ち上げ◎科目名の登録◎人名リストの出力◎得点リストの出力◎得点順位◎平均・最高・最低得点◎成績順に並べかえ◎グラフィック画面の保存◎顧客管理プログラム◎顧客名の登録◎売り上げ金額の入力◎入金の登録◎売り上げ金額の合計◎アイウエオ順に並べかえ◎スケジュール管理プログラム◎予定の入力◎予定の自動作成◎時刻早送り◎割り込みプログラム

PC-8801mkIIFR|MR
データファイル入門[1]

発行――――1986年1月25日
編者――――ナツメ出版企画
発行者―――田村正隆
発行所―――株式会社ナツメ社
郵便番号101
東京都千代田区神田神保町1-52
電話〈03〉291-1257
振替　東京3-58661
〈落丁・乱丁本はお取り替えします〉

定価――――**1600円**

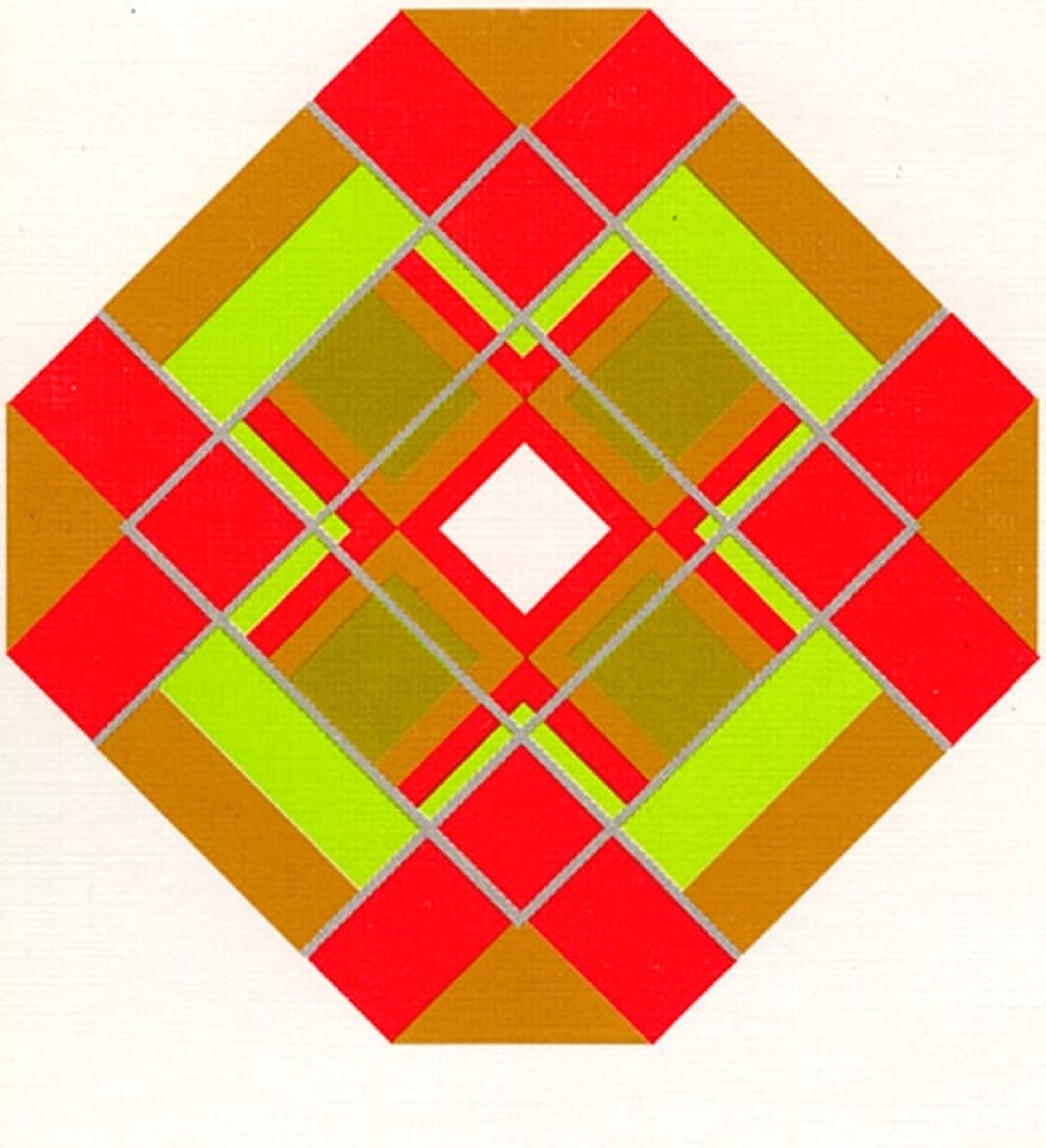
PC-8801mkⅡFR|MR

データファイル入門[1]

ナツメ社

定価＝1600円

[目次より]



ISBN4-8163-0566-1 C2054 ¥1600E